A Text Book for Computer Software & Hardware

XI Computer Science

FOR XI STD. BIFOCAL COMPUTER SCIENCE

COMPUTER SOFTWARE (PAPER I)
COMPUTER HARDWARE (PAPER II)
PRACTICAL (PAPER I & II)

R. D. SUPEKAR

B.Sc. (Physics-Elect.) D.E.R.E., M.C.M. D.I.T

HOD Vocational Department Junior College
Lecturer in Electronics and Computer Science,
Teacher in Charge, Vocational Electronics Department,
S.P. College, Pune.
Visiting Faculty Member,
Prin. N.G. Naralkar Institute of Career Development and Research, Pune.

MRS. VAISHALI R. GHULE

B.E. (Elect.), Sun Certification, USA

Lecturer in Computer Science, S.P. College, Pune.
Visiting Faculty Member, Prin. N.G. Naralkar Institute of Career Development and Research, Pune.

MRS. TARAKA GHAMANDE

B.E. (Elect.), DAC

Lecturer in Computer Science,
Sinhgad College of Science
Junior College, Pune.

VILAS KARMUDE
B.Sc. M.C.A A.D.C.A.

Lecturer in Computer Science,
Ramnivas Ruia Jr. College, Matunga, Mumbai.

MAYURESHWAR PRAKASHAN PUNE

Mob: 94 220 180 66

1. NUMBER SYSTEMS

INTRODUCTION

Digital electronics is one of the important branches; it is related with computer circuits, digital systems, digital communications, counters etc. We shall study different number systems, their inter-conversions and binary arithmetic. This topic is important because it includes illustrations of different machine codes, binary arithmetic and their conversions called as encoding and decoding methods. After completing this chapter you will be able to convert any given number into any computer code and able to add, subtract, multiply or divide using binary numbers.

1.1 BINARY NUMBER SYSTEM

The number system, which we use, is a decimal number system; we use basic ten symbols or numbers from 0 to 9. But in digital circuits; it is very difficult to use these numbers directly because of ten different levels, size required for the circuit and reliability of the circuit. Therefore, a new number system is developed so that only two levels are required for the operation. This new number system is known as ***"Binary Number System"*** where Bi -means two, and two symbols are used as **'0'** and **'1'**

Advantages of binary code

(1) Only two voltage levels are used to represent these two numbers.

(2) Different logic can be applied behind these two numbers such as if we take lamp, meter, electric signal to represent these numbers then.

For Example,

Logic	**0**	**1**
Lamp	OFF	ON
Voltmeter	0V	+5V and so on.

(3) It is very easy to perform mathematical operations like addition, subtraction with these numbers.

(4) Conversion of binary to decimal and decimal to binary is also easy and inter-convertible.

(5) Binary Number system is more accurate and readable number system.

Positive and negative logic system

In positive logic, the higher voltage level indicates level 1 and in negative logic it indicates level 0. e.g. if levels are +5V and 0V then in positive logic +5V indicates level

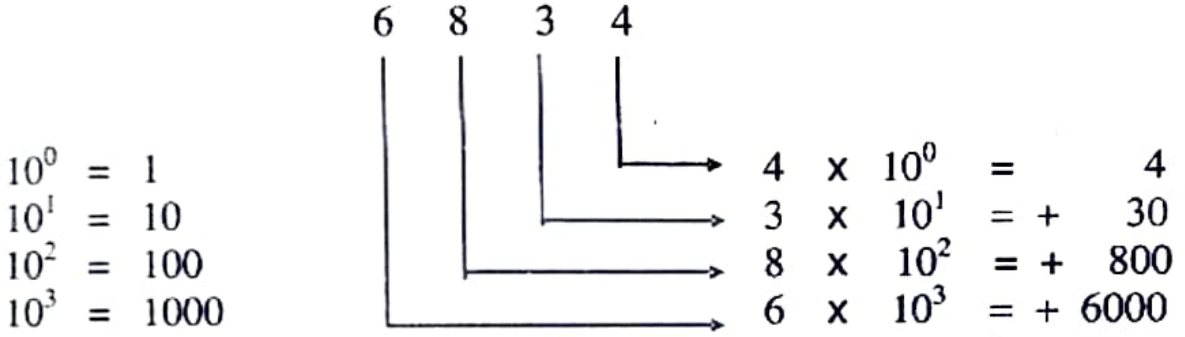
1 and in negative logic 0V indicates level 1. In case of lamp, in positive logic; the level 1 logic is lamp ON and in negative the logic level 1 logic is lamp OFF.

RADIX

Radix is a number, which indicates the base of specific number system. This number represents total symbols of the system and weightage or power of each digit.

For example,

1) The radix for decimal number system is '10' because total symbols are 10 from 0 to 9. Similarly, the weightage of each digit is in power of 10. Suppose number is six thousand eight hundred and thirty-four then it is represented as $(6\ 8\ 3\ 4)_{10}$. The subscript '10' indicates that the number is a decimal number. Its value is calculated as follows



2) In this way if we take binary number system then its radix is '2' because symbols are only two 0 and 1. A number in binary number system is represented as $(101011)_2$ Multiplying each 0 and 1 by power of 2 and adding them can calculate its value. We shall discuss this conversion in the next topic. The radix for other number systems are given in table (1.1)

Number System	Radix	Symbols
1) Decimal Number System	10	0,1,.....8,9
2) Binary Number System	2	0 and 1
3) Octal Number System	8	0,1,.....7
4) Hexadecimal Number System	16	0,1,.....8,9,A,B,...F

Table (1.1)

Binary Counting

As we know the decimal counting is sequence like 0,1,2,3 ---- 9, 10, 11 ---- 99,100---- etc. To get binary counting take the decimal numbers in sequence and take those numbers in sequence, which contain only 0 and 1.

0,1,2 ---- 10, 11, --- 100, 101, 102 --- 110, 777, ---- 1000, 1001, 1003 ---

Binary Number	Decimal Value	Binary Number	Decimal Value
0	0	1 1 0	6
1	1	1 1 1	7
10	2	1 0 0 0	8
11	3	1 0 0 1	9
100	4	1 0 1 0	10
101	5	1 0 1 1	11

Table (1.2)

1.2 BINARY TO DECIMAL CONVERSION

To convert a binary numbers into decimal number follow the given procedure

Step 1 : Write the given binary Number

Step 2 : Write the binary weightage below each number

Decimal point

Binary Weightage	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
Decimal Value	128	64	32	16	8	4	2	1	0.5	0.25	0.125

Table (1.3)

Step 3 : Cancel the weightage, which is placed below zero because any number multiplied by zero is zero.

Step 4 : Add the remaining numbers.

Examples :

1) Convert $(1001)_2$ in to decimal no.

Step 1: 1 0 0 1

Step 2 : 8 + 4 + 2 + 1

Step 3: 8 + ~~4~~ + ~~2~~ + 1

Step 4 : 8 + 1

$\therefore (1001)_2 = (9)_{10}$

(Verification)

1 0 0 1

$1 \times 2^0 = 1$

$0 \times 2^1 = + 0$

$0 \times 2^2 = + 0$

$1 \times 2^3 = + 8$

$= 9$

Refer table (1.2) 1001 is equivalent to decimal nine.

2) Convert $(110011)_2$ into decimal

$= 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$

$= 32 + 16 + ~~8~~ + ~~4~~ + 2 + 1$

$= (51)_{10}$

$\therefore (110011)_2 = (51)_{10}$

3) Convert $(1100110)_2$ into decimal

$$= 64 + 32 + 16 + 8 + 4 + 2 + 1$$
$$= (102)_{10}$$

4) Convert $(11101)_2$ into decimal

$$= 16 + 8 + 4 + 2 + 1$$
$$= 29$$
$$\therefore \quad (11101)_2 = (29)_{10}$$

5) If $(100001)_2 = (X)_{10}$ find X

$$32 + 16 + 8 + 4 + 2 + 1$$
$$X = 33$$
$$\therefore (100001)_2 = (33)_{10}$$

Fractional binary numbers

To convert the mixed binary numbers containing integers and fractions follow the same procedure and make use of the table (1.3). Let us solve two examples

1) Convert $(10101.101)_2$ into decimal

$$16 + 8 + 4 + 2 + 1 . + 0.5 + 0.25 + 0.125$$
$$= (21.625)_{10}$$

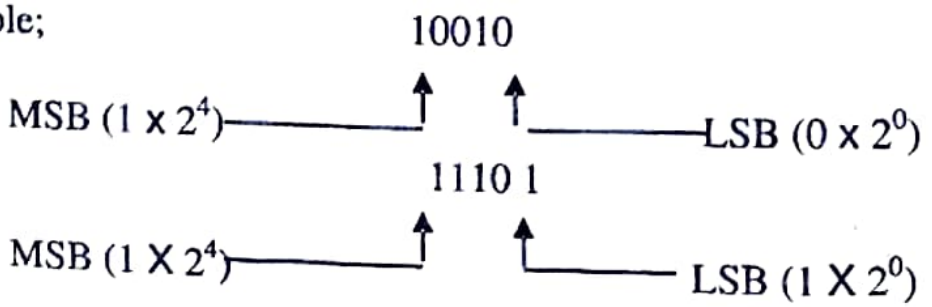
2) Convert $(1001.011)_2$ into decimal

$$8 + 4 + 2 + 1 . 0 + 0.25 + 0.125$$
$$= (9.375)_{10}$$

1.3 DECIMAL TO BINARY CONVERSION

Let us see now, how a decimal number is converted into its binary equivalent number. Dividing the given number by two and taking only remainders do this. This method of divide by two is known as "**double - dabble method**". Before that; let us see what is MSB and LSB. MSB represents "**Most Significant Bit**" that represents the first number in given number, which has got maximum weightage. LSB means the "**Least Significant Bit**" which represents the last number which has got lowest weightage.

For Example;



Double Dabble Method

To convert decimal number into binary number divide the given number by '2' till you get '1' as shown in the following examples

1.

29	÷ 2	remainder	29 - 28	= 1	↑ LSB
14	÷ 2	remainder	14 - 14	= 0	
7	÷ 2	remainder	7 - 6	= 1	
3	÷ 2	remainder	3 - 2	= 1	
1	MSB				

$(29)_{10} = (11101)_2$

It can be verified by reverse process

1 1 1 0 1

$16 + 8 + 4 + \not{2} + 1 = 29$

2. Convert $(35)_{10}$ into binary

LSB		
1	35 - 34	(35 ÷ 2 = 17)
1	17 - 16	(17 ÷ 2 = 8)
0	8 - 8	(8 ÷ 2 = 4)
0	4 - 4	(4 ÷ 2 = 2)
0	2 - 2	(2 ÷ 2 = 1)
	1	← MSB

$\therefore (35)_{10} = (100011)_2$

3. Convert $(102)_{10}$ into binary.

0	102- 102
1	51 - 50
1	25 - 24
0	12 - 12
0	6 - 6
1	3 - 2
	1

$\therefore (102)_{10} = (1\ 100110)_2$

4) Convert $(88)_{10}$ into binary

0	88 - 88
0	44 - 44
0	22 – 22
1	11 – 10
1	5 – 4
0	2 – 2
	1

$\therefore (88)_{10} == (1011000)_2$

5) Convert $(69)_{10}$ into binary

1	69 - 68
0	34 - 34
1	17 - 16
0	8 - 8
0	4 - 4
0	2 - 2
	1

$\therefore (69)_{10} = (1000101)_2$

Fractional numbers

To convert a fractional numbers in to decimal multiply each bit by 2 and take integer value out as shown in the following examples. In these examples integers can be converted by double-dabble method (not shown) refer the above examples.

1) Convert $(88.625)_{10}$ into binary

.625 x 2 = 1.25 — 1

.25 x 2 = 0.5 — 0

.5 x 2 = 1.0 — 1

$\therefore (88.625)_{10} = (1011000.101)_2$

2) Convert $(69.375)_{10}$ into binary

.375 x 2 = 0.75 — 0

.75 x 2 = 1.50 — 1

.5 x 2 = 1.0 — 1

$\therefore (69.375)_{10} = (1000101.011)_2$

(Note that in case of odd fractional members find the binary numbers up to 4 digits or 5 digits to use approximation)

1.4 HEXADECIMAL NUMBER SYSTEM

In this system hex means six and decimal indicates ten, total sixteen symbols involves from 0 to 9 and A to F. Its radix is 16, each number is expressed in the power of 16. The following symbols indicate their decimal values.

0 1, 2.....9,	A	B	C	D	E	F
	↓	↓	↓	↓	↓	↓
	10	11	12	13	14	15

Hex to decimal conversion

To convert a given hexadecimal number into decimal each number is multiplied by 16^n. i.e.

1) $(9\ A\ E)_{16} = (\quad)_{10}$

9 A E

$14 \times 16^0 = 14$

$10 \times 16^1 = +\ 160$

$9 \times 16^2 = +\ 2304$

$= 2478$

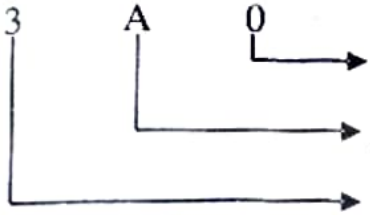
$(9\ A\ E)_{16} = (2478)_{10}$ .....Ans.

2) $(2\ C\ 6\ E)_{16}$ (Alternative Method)

$= \underset{(2 \times 16^3)}{2} + \underset{(C \times 16^2)}{C} + \underset{(6 \times 16^1)}{6} + \underset{(E \times 16^0)}{E}$

= (2 x 4096) + (12 x 256) + (6 x 16) + (14 x 1)
= 8192 + 3072 + 96 + 14
= $(11374)_{10}$

3) $(3\ A\ 0)_{16} = (\ ?\ )_{10}$



$= 0$
$= + 160$
$= + 768$
$= 928$

$(3\ A\ 0)_{16} = (928)_{10}$ .....Ans.

4) $(7FF.8)_{16} = (\ )_{10}$

= 7 F F . 8
= (7×16^2) + $(F \times 16^1)$ + $(F \times 16^0)$. (8×16^{-1})
= 1792 + 240 + 15 + 0.5
= $(2047.5)_{10}$

5) $(2AD.2B)_{16} = (\ )_{10}$

= 2 A D . 2 B
= (2×16^2) + $(A \times 16^1)$ + $(D \times 16^0)$. (2×16^{-1}) (11×16^{-2})
= 512 + 160 + 13 + 0.125 + 0.04296
= $(685.1679)_{10}$

Decimal to Hexadecimal conversion

Convert the given decimal number into hexadecimal by using hex-dabble method, like double dabble method where the given number is divided by 16.

1) $(928)_{10}$ =

0	928
10	58
	3

$(928)_{10} = (3A0)_{16}$

2) $(2478)_{10}$

14	2478
10	154
	9

$(2478)_{10} = (9A\ E)_{16}$
Where 14 = E and 10 = A

Hex to Binary and Binary to Hex conversion

To convert a hexadecimal number in to binary number, convert each hex digit in to a 4-bit binary code just like 8421 BCD. On the other hand, to convert a binary number in to Hexadecimal number make the group of 4-bits and find their Hex values refer the following examples

1) Convert $(5\ A\ D)_{16}$ in to binary
= 5 A D
= 0101 1010 1101

2) Convert $(C9F.2A)_{16}$ in to binary
= C 9 F . 2 A
= 1100 1001 1111. 0010 1010

3) Convert $(110100011111)_2$ in to Hex
= 1101 0001 1111
= D 1 F

4) Convert $(11110010.1110)_2$ in to Hex
= 1111 0010 . 1110
= F 2 . E

1.5 OCTAL NUMBER SYSTEM

In this system eight symbols from 0 to 7 are used and its radix is 8.
To convert a octal number in to decimal follow the given table.

Decimal point

Octal Weightage	8^4	8^3	8^2	8^1	8^0	8^{-1}	8^{-2}
Decimal Value	4096	512	64	8	1	0.125	0.015625

1) $(2\ 5\ 7)_8 = (\quad)_{10}$

2 5 7

$7 \times 8^0 = 7$

$5 \times 8^1 = +\ 40$

$2 \times 8^2 = +\ 128$

$= 175$ $(2\ 5\ 7)_8 = (1\ 7\ 5)_{10}$....Ans.

2) $(1256)_8 = (\quad)_{10}$ (Alternative Method)

	1		2		5		6
=	(1×8^3)	+	(2×8^2)		$+(5 \times 8^1)$	+	(6×8^0)
=	(1×512)	+	(2×64)	+	(5×8)	+	(6×1)
=	512	+	128	+	40	+	6
=	$(686)_{10}$						

3) $(730)_8 = (\quad)_{10}$

7 3 0

$0 \times 8^0 = 0$

$3 \times 8^1 = +\ 24$

$7 \times 8^2 = +\ 448$

$= 472$

$(730)_{16} = (472)_{10}$ .....Ans.

Decimal to Octal conversion

Convert the given decimal number into Octal, like double dabble method where the given integer number is divided by 8 and for fractional number multiply by 8.

1) $(175)_{10}$ =

7	175
5	21
	2

2) $(2470)_{10}$

6	2470
4	308
6	38
	4

$(175)_{10} = (257)_8$ .....Ans.

$(2470)_{10} = (4646)_8$.....Ans.

Octal to Binary and Binary to Octal Conversion

To convert a Octal number in to binary number, convert each hex digit in to a 3-bit binary code. On the other hand, to convert a binary number in to Octal number make the group of 4-bits and find their decimal values refer the following examples

1) Convert $(523)_8$ in to binary

$= 5 \quad 2 \quad 3$

$= 101\ 010\ 011$

2) Convert $(704.01)_8$ in to binary

$= 7 \quad 0 \quad 4 . 0 \quad 1$

$= 111\ 000\ 100 . 000\ 001$

3) Convert $(110100011111)_2$ in to Octal

$= 110\ 100\ 011\ 111$

$= 6 \quad 4 \quad 3 \quad 7$

4) Convert $(110100101.110011)_2$ in to Octal

$= 110\ 100\ 101 . 111\ 011$

$= 6 \quad 4 \quad 5 . 7 \quad 3$

Hex to Octal and Octal to Hex Conversion

To convert a Hex number in to Octal first convert **Hex in to Binary** and then make groups of 3-bits from LSB add zeros on left and right side if required. After making the groups convert each 3-bit binary in to Octal equivalent.

1) $(5CA)_{16} = (?)_8$

$= 5 \quad C \quad A$

$= (0101\ 1100\ 1010)_2$

2) $(3B.2E)_{16} = (?)_8$

$= 3 \quad B \quad . \quad 2 \quad E$

$= (0011\ 1011 . 0010\ 1110)_2$

$= 010\ 111\ 001\ 010$
$= 2\ \ 7\ \ 1\ \ 2$
$(5\ C\ A)_{16} = (2712)_8$

$= 000\ 111\ 011.\ 001\ 011\ 100$
$= 0\ \ 7\ \ 3\ .\ 1\ \ 3\ \ 4$
$(3B.2E)_{16} = (073.134)_8$

To convert a Octal number in to Hex first convert **Octal in to Binary** and then make groups of 4-bits from LSB add zeros on left and right side if required. After making the groups convert each 4-bit binary in to Hex equivalent refer the following examples

3) $(537)_8 = (?)_{16}$
$= 5\ \ 3\ \ 7$
$= (101\ 011\ 111)_2$
$= 0001\ 0101\ 1111$
$= 1\ \ 5\ \ F$
$(5\ 3\ 7)_8 = (15F)_{16}$

2) $(53.21)_8 = (?)_{16}$
$= 5\ \ 3\ .\ 2\ \ 1$
$= (101\ 011\ .\ 010\ 001)_2$
$= 0010\ 1011.\ 0100\ 0100$
$= 2\ \ D\ .\ 4\ \ 4$
$(53.21)_8 = (2D.44)_{16}$

1.6 BCD, ASCII and EBCDIC CODES

In binary, for higher decimal numbers it becomes a long chain of 0 & 1, to avoid this, different BCD codes are used to represent decimal numbers. In computer arithmetic circuits 8421 BCD is a most common code.
In BCD code, each decimal number is represented by a 4-bit code.

e. g. 5 → 0101 ; 3 → 0011 ; 9 → 1001

The zeros on left of 8421 BCD can be omitted. To convert a BCD number into decimal back, the process is exactly opposite, by making group of 4 bits from the left and converting each 4 - bit number into decimal number.

e. g. 100100111001_{BCD}
$= 1001\ \ 0011\ \ 1001$
$= 9\ \ 3\ \ 9$

There are some other codes like 5421 BCD, 5311 BCD etc. But the most common method is 8421 BCD.

Advantages of BCD

1. The length of number is short.
2. Easy to convert to and from BCD
3. Suitable for hexadecimal addition.

ASCII and EBCDIC CODES

These codes are normally used for computer keyboard; ASCII code uses either 7- bit code or 8-bit code to represent the numbers and the characters. ASCII stands for American Standard Code for Information Interchange. EBCDIC code is a similar 8- bit code it is developed by IBM which is obtained by extending six bit BCD code. It is Extended BCD Interchange Code. e.g. In ASCII the code for CAT is easily obtained by writing the binary code for C,A and T

C=11 0011 A = 11 0001 T= 01 0011 (Refer ASCII 7 –bit Chart)

CAT =(11 0001 11 0001 01 0011)

1.7 BINARY ADDITION AND SUBTRACTION

In computer circuits arithmetic logic unit performs basic arithmetic operations by performing addition only like subtraction by addition, multiplication by addition etc.

Let us see basic addition and subtraction rules in binary.

Rules	Addition	Subtraction
1.	0 + 0 = 0	0 – 0 = 0
2.	0 + 1 = 1	0 – 1 = 11 (difference1and borrow 1)
3.	1 + 0 = 1	1 – 0 = 1
4.	1 + 1 = 10 (Carry1 sum 0)	1 – 1 = 0
5.	1 +1 + 1 = 11 (Carry1 sum1)	

e. g.

1 2		1 1 0 0	(Verification)
+1 4		+1 1 1 0	1 1 0 1 0
	Carry ⟶	1 1	= 16 8 4 2 0
2 6		1 1 0 1 0	= 16 + 8 + 2 = 26

The way to verify the binary addition and subtraction, first find the decimal equivalent of the given binary number and then check the addition with the final binary number.

e.g.

1 3	1 1 0 1	First column 1 – 0 = 1
- 1 0	-1 0 1 0	Second column 0 – 1 = 11
3	0 0 1 1	i.e. difference 1 and borrow 1
		Third column 1-(0 + 1) = 0
		Fourth column 1 – 1 = 0

The above method discussed is not suitable for signed numbers, 1's or 2's complement method is used for signed numbers.

1.8 1'S AND 2'S COMPLEMENT FOR SUBTRACTION

These two methods can perform subtraction by addition. Similarly, they can generate signed numbers like + 2 , - 5 etc. After subtraction one can easily verify the difference with sign.

Before we learn subtraction let us find 1's and 2's complement of a given number (A)

'$\overline{A}$ 'denotes 1's complement and it is obtained by replacing 0 by 1 and 1 by 0.

e. g. If A = 1010 then $\overline{A} = 0101$

2's complement is the number that results when we add 1 to the 1's complement. It is denoted by A′ 2's complement = (1's complement) + 1

$$A' = \overline{A} + 1$$

Examples1: Find 1's complement of 101011 and 100110

Solution: i) A = 101011 $\overline{A}$ = 010100 (1's complement)

ii) A = 100110 $\overline{A}$ = 011001 (1's complement)

Examples2: Find 2's complement of 111010 and 110100

Solution: i) A = 111010 $\overline{A}$ = 000101 (1's complement)

A' =000101 +1 = 000110 (2's complement)

i) A = 110100 $\overline{A}$ = 001011 (1's complement)

A' =001011 +1 = 001100 (2's complement)

The procedure to perform subtraction A - B, follow the following procedure.

	1's complement subtraction	**2's complement subtraction**
Step 1	Find binary of A	Find binary of A
Step 2	Find 1's complement of B	Find 2's complement of B
Step 3	Add step 1 with step 2	Add step 1 with step 2
Step 4	i) If carry 1 is present answer is +ve make End around carry.	If carry 1 is present answer is +ve Delete the carry.
	ii) If carry is absent answer is –ve find 1's complement	If carry is absent answer is –ve find 2's complement.

Examples :

1. Perform 12 – 9 and 9 – 12 by using 1's complement method.

Solution : (12 – 9)

Step 1 Binary of 12 is 1 1 0 0

Step 2 Binary of 9 is 1 0 0 1

1's complement is 0 1 1 0

Step 3

$$\begin{array}{rr} & 1\,1\,0\,0 \\ + & 0\,1\,1\,0 \\ \hline \text{Carry present} \rightarrow & \boxed{1}\,0\,0\,1\,0 \end{array}$$

Step 4 Answer is +ve end around carry

$$\begin{array}{l} \not{\boxed{1}} \\ 1\,0\,0\,1\,0 \\ \quad \rightarrow +1\ (\text{EAC}) \\ \hline \text{Ans.}\quad 0\,0\,1\,1 \quad = \quad (+3) \end{array}$$

(9 – 12)

Binary of 9 is 1 0 0 1

Binary of 12 is 1 1 0 0

1's complement is 0 0 1 1

$$\begin{array}{rr} & 1\,0\,0\,1 \\ + & 0\,0\,1\,1 \\ \hline \text{Carry absent} \rightarrow & \boxed{\ }\,1\,1\,0\,0 \end{array}$$

Answer is –ve

1's complement of 1 1 0 0 is

0 0 1 1 = (-3)

2. Perform 14 – 9 and 9 – 14 by using 2's complement method.

Solution : (14 – 9)

Step 1 Binary of 14 is 1 1 1 0

Step 2 Binary of 9 is 1 0 0 1

2's complement is 0 1 1 1

Step 3

$$\begin{array}{rr} & 1\,1\,1\,0 \\ + & 0\,1\,1\,1 \\ \hline \text{Carry present} \rightarrow & \boxed{1}\,0\,1\,0\,1 \end{array}$$

Step 4 Answer is +ve delete the carry

$$\begin{array}{l} \not{\boxed{1}}\,0\,1\,0\,1 \\ \text{Ans.}\quad 0\,1\,0\,1 \quad = (+5) \end{array}$$

(9 – 14)

Binary of 9 is 1 0 0 1

Binary of 14 is 1 1 1 0

2's complement is 0 0 1 0

$$\begin{array}{rr} & 1\,0\,0\,1 \\ + & 0\,0\,1\,0 \\ \hline \text{Carry absent} \rightarrow & \boxed{\ }\,1\,0\,1\,1 \end{array}$$

Answer is –ve

2's complement of 1 0 1 1 is

0 1 0 1 = (-5)

1.9 BINARY MULTIPLICATION AND DIVISION

These two Arithmetic operations in binary are easily performed just like simple decimal multiplication and division. Consider the following illustrative examples.

i) $(101 \times 110)_2 = (5 \times 6)_{10}$

```
  1 0 1
 x1 1 0
-------
    0 0 0
+ 1 0 1 x
+1 0 1 x x
-----------
1 1 1 1 0
```

$= (30)_{10}$

ii) $(1101 \times 111)_2 = (13 \times 7)_{10}$

```
   1 1 0 1
   x 1 1 1
-----------
     1 1 0 1
+  1 1 0 1 x
+1 1 0 1 x x
-------------
1 0 1 1 0 1 1
```

$= (91)_{10}$

iii) $(1100 \div 10)_2 = (12 \div 2)_{10}$

```
   1 1 0
  11 00
 -10
  ----
    10
   -10
   ---
    0 0
```

Ans: $(110)_2 = (6)_{10}$

iv) $(11001 \div 101)_2 = (25 \div 5)_{10}$

```
   1 0 1
  -----
  11001
- 101
  -----
  00101
   -101
   ----
    000
```

Ans: $(101)_2 = (5)_{10}$

SOLVED EXAMPLES

1. Convert the following.

i) $(7AB)_{16} = (\ ?\ )_{10}$

7 A B

$11 \times 16^0 = 11$

$10 \times 16^1 = +\ 160$

$7 \times 16^2 = +\ 1792$

$= 1963$

$(7AB)_{16} = (1963)_{10}$ .....**Ans.**

ii) $(1001\ 10000\ 0001)_{BCD} = (\ ?\ )_{10}$

making group of 4 bits

$(1001) = (9)_{10}$ $(1000) = (8)_{10}$ $(0001) = (1)_{10}$

$(1001\ 1000\ 0001)_{BCD} = (981)_{10}$ .....**Ans**

2, Convert the following:

i) $[11001.101]_2 = [\ ]_{10}$

$=1\ \ 1\ 0\ 0\ 1\ .\ 1\ \ 0\ \ 1$

$=16\ 8\ 4\ 2\ 1\ .\ 0.5\ 0.25\ 0.125$

$=16+8+1\ .\ 0.5+0.125$

$=[25.625]_{10}$

ii) $[110101.110001]_2 = [\ ]_{16}$

$=110101.110001$

$=0011\ 0101.\ 1100\ 0100$

$=3\ \ \ 5\ \ .C\ \ \ 4$

$=[35.C4]_{16}$

iii)$[2AF]_{16} = [\ ]_2$ Convert it into BCD

$= 2\ \ \ A\ \ \ F$

$= 0010\ \ 1010\ \ 1111$ $\quad \therefore [2AF]_{16} = [0010\ 1010\ 1111]_2$

3. Perform the following operation using 2's complement method use 8-bit representation of numbers $(52)_{10} - (65)_{10} = (?)_2$

Solution : 8-bit binary equivalent of 52 = 00110100 ..... (i)

8-bit binary equivalent of 65 = 01000001

1^S complement of 65 = 10111110

2^S complement of 65 = 10111110 + 1

= 10111111 ..... (ii)

52-65 = 00110100

+10111111

No carry ⟶ □11110011

∴Answer is negative

Find $2'^S$ complement of the result

$\underline{A} = 11110011$

$\overline{A} = 00001100$

Ans : $A' = 00001100+1 = (00001101)_2 = (-13)_{10}$

QUESTIONS

1. Select the correct answer.

i) The BCD equivalent of a decimal number 88 is.....

a) 0010 b) 0001 c) 1000 1000

ii) The decimal equivalent of a hexadecimal number 2 A is.....

a) 42 b) 62 c) 82 d) 17

iii) The Radix of the octal system is.....

a) 2 b) 16 c) 10 d) 8

iv) The binary equivalent of hexadecimal number $(B3)_{16}$ is……

a) 10110011 b) 11011011 c) 0011 d) 00010001

v) In positive logic system '1' represents……

a) The more negative of two voltage levels. b) zero voltage.

c) The more positive of two voltage levels. d) Negative one volt.

vi) 2's complement of $(12)_{10}$ is…..

a) 0001, b) 0010 c) 0011 d) none of these

vii) In binary numbers if the last bit is 1 then it is a……number

a) odd b) even c) none of these

viii) $(11)_2 + (11)_2 + (11)_2 = ...$

a) $(111)_2$ b) $(1011)_2$ c) $(1001)_2$ d) $(1111)_2$

ix) The equivalent decimal number of maximum highest binary number of length one byte is …….

a)128 b) 127 c) 255 d) 256

Ans. i) 1000 1000 ii) 42 iii) 8 iv) 1011 0011 v) The more +ve of two voltage levels vi) none of these vii) odd viii) $(1001)_2$ ix)255

Problems For Practice

1. Convert the following decimal numbers into binary

   a) 38 b) 25.5 c)10.625

2. Encode the following numbers into BCD code

   a) 128 b) 579 c) ABC

3. Convert the following binary numbers into decimal number

a) 11001.101 b) 0.110 c) 101100.001

4. Convert the following Hex into Decimal.

a) 2E9 b) 09F7 c) 12C

5. Convert the following Hex into Binary

a) 3A5E b) 0A3.D9 c) 1B5.C

6. Convert the following Hex into Octal.

a) 2F7 b) BAB c) ABC.4D

7. Convert the following octal numbers into binary

a) 736 b) 574.321 c)116.54

8. Perform the following:

a) $(10111 \cdot 101)_2 = (\ )_{16}$ b) $(91)_{10} = (\ )_2$ c) $(594)_{10} = (\ )_{BCD}$

9. Convert the following :

a) $[100]_{10} = [?]_{16}$ b) $(DAD)_{16} = [?]_{10}$ c) $[BCD]_{16} = [?]_2$ d) $[10101 \cdot 11]_2 = [?]_{10}$

10. Convert the following

a) $(ABC \cdot D)_{16} = (\quad)_{10}$ b) $(10101101 \cdot 11)_2 = (\quad)_{10}$

c) $(11011010 \cdot 1001)_2 = (\quad)_{16}$

Ans: 1) a) $(100110)_2$ b)$(11001.1)_2$ c) $(1010.101)_2$

2) a) (0001 0010 1000) b) (0101 0111 1001) c)(1010 1011 1100)

3) a) $(25.625)_{10}$ b) $(.75)_{10}$ c) $(44.125)_{10}$

4) a) $(745)_{10}$ b) $(2551)_{10}$ c) $(300)_{10}$

5) a) $(0011\ 1010\ 0101\ 1110)_2$ b) $(0000\ 1010\ 0011.1101\ 1001)_2$

c)$(0001\ 1011\ 0101.1101)_2$

6) a) $(1367)_8$ b) $(5653)_8$ c) $(5274.232)_8$

7) a) $(111\ 011\ 110)_2$ b) $(101\ 111\ 100\ .\ 011\ 010\ 001)_2$ c) $(001\ 001\ 110\ .\ 101\ 100)_2$

8) a) $(17.A)_{16}$ b) $(1011011)_2$ c) $(0101\ 1001\ 0100)_{BCD}$

9) a) $(64)_{16}$ b) $(3510)_{10}$ c) $(1011\ 1100\ 1101)_{BCD}$ d) $(21.75)_{10}$

10) a) $(2748.8125)_{10}$ b) $(173.75)_{10}$ c) $(DA.9)_{16}$

Solve the following:

1. What do you mean by

   a) 1's complement b) 2's complement explain with suitable examples.

2. Perform the following using binary arithmetic.

   a) 12-14 b) 23+25 c) 111101 + 110011

3. Subtract the following by 1's complement method:

   a) 1110 – 1011 b)1101 – 1110

4. Perform the following by Two's complement method use 8-bit representation $(40)_{16}$ -$(64)_{16}$

5. Explain the following

   a) Radix b) BCD 8421 Code

   c) 1's complement d) + ve and – ve logic system

6. Find X if i) $(2AC)_{16} = (X)_2$ ii) $(11110.1101)_2 = (X)_{10}$

7. Find 12- 9 and 8 –11 by using 1's complement method.

8. Explain 2's complement subtraction with suitable examples.

9. Perform 27 +29 by using binary arithmetic.

10. Convert the following decimal number into binary

    a) 38.625 b) 45.875 by double dabble method.

12. Convert the following binary number into decimal number

    a) 11001.101 b) 0.110

13. Perform the following conversions

    i) $(253)_{16} = (\quad)_{10}$ ii) $(4AF)_{16} = (\quad)_{10}$ iii) $(2AB.3E)_{16} = (\quad)_2$

    iv) $(1010100.110)_2 = (\quad)_{10}$ v) $(253)_8 = (\quad)_{16}$ vi) $(253.56)_8 = (\quad)_{10}$

    vii) $(110011100.1010)_2 = (\quad)_{16}$ viii) $(7E5.4B)_{16} = (\quad)_8$

14. Multiply 6 x 4 and 9 x 7 by using binary numbers.

15. Perform 15 ÷ 5 and 24 ÷ 6 by using binary numbers.

❑❑❑

2. PROGRAM ANALYSIS

INTRODUCTION

Programming is the software concept where it is the playback of computer job. Computer person should be perfect in programming. The results of any computer job, printout, layout, results, data processing; all these tasks are basically depend upon the programming technique and accuracy of programming. Your program tells computer to do particular job in a desired manner. **Program Development** is not so simple it requires number of systematic working steps to make it easy simple and perfect. This chapter explains different concepts and illustrative examples to get the knowledge of program designing aspect. Program development is a skill, it requires logic. Any problem, which is to be computerised, must be properly analysed and organised. It is to be always planned because computer is very precious and one can't afford computer by experimentally on methods of problem solving.

Any problem, which has to be converted in computer program, must be carefully analysed. Problem analysis consists in finding or deriving the procedural steps involved in getting the solution to a problem. That step must come within the computer capabilities.

e.g. Problem statement

Add 2 numbers and display the result

1) Take 2 numbers suppose a and b as input.
2) Add 2 numbers.
3) Store the result in c.
4) Display value of c.

In this way above simple problem is divided into 4 steps.

For analysis various methods are used. These methods are

1) Flowchart, 2) Algorithm 3) Pseudocode.

3.1 PROGRAM DEVELOPMENT

Program is a list of instructions in specific language to tell the computer to do the job in sequence. Before writing a program user has to define problem. Problem is first analyzed. Actually problem solution gets converted into program. But for the conversion there is a process of programming. Programming is a process for development of program, for a specific problem.

Whenever any problem has to be converted in to a program, then it's process has following steps:

1) Problem analysis
2) Design
3) Coding
4) Testing
5) Documentation

1) Problem Analysis

You should have complete definition of problem and the requirements should be clear before writing the solution. Generally every program needs input, output, storage and process.

Input: Inputs to be supplied for the program.

Output: Outputs means in what form output is required. It represents the computer solution to the given problem.

Storage: If output is there, the storage is required i.e. o/p has to be stored in some form. Input parameters also required to be stored.

Process: Now the events should be executed. What is its sequence?

In general, any problem given has to be analyzed. That problem is divided into simple steps leading to solution or a problem. We have to make a plan of problem solution and accordingly program is written.

2) Design

In design phase, generally solution plan is divided into small modules and then every module is designed separately. Sometimes some developers design easy modules and then design the remaining modules. Obviously according to development strategy, various designing methods are available; programmer can use those methods also. Structured programming and object oriented programming has various design methods. User can select one out of them.

3) Coding

It is the process of conversion of a program design into programming language.

4) Testing

After coding, programmer has to test the program to see whether it is executing correctly or not. If it is having any problems, errors then user has to see it and correct it, verification is also important.

5) Documentation

The written information of program that has to be maintained for developer as well as user is document of program. So there are two types of documentation.

a) Developer's documentation: In this type all information related to program codes, designing methods, testing methods are maintained.

b) User's documentation: In this type information related to operating program, input parameters, and output parameters is maintained.

Programmer documentation:

1) Requirement and specifications of document
2) Design documents: Copy of flowcharts, algorithm, pseudocode.
3) Copy of source code program.
4) Testing reports with all types of possible inputs.
5) Output reports with all types of possible outputs for a program.

User documentation is user manual generally i.e. how to operate the program.

3.2 PROBLEM ANALYSIS TOOLS

There are various tools used for problem analysis. As we have already said for analysis or for defining logic we have to divide program into steps, then you can show overall logic using following methods:

a) Algorithm
b) Flowchart
c) Pseudocode

a) Algorithm

Algorithm is a logical sequence of steps prepared for solving problem. It is a tool to write the job in simple steps in simple English language. It is not programming language or program. It guides the programmer to prepare the program.

Example1: Algorithm to exchange the value of two variables.

Suppose if A and B are the two variables

Algorithm:

1) Store the value of A in X
2) Transfer B value to A
3) Transfer X to B

Example2: To Develop the program to drive a motorcycle:

Step I – Understand the mechanism and facilities provided with the motorcycle.

Step II – Study the sequence of operation.

Step III – Algorithm

1) Insert the ignition key
2) Apply the kick or starter
3) Remove the stand
4) Put the motorcycle into first gear by holding clutch
5) Release the clutch by increasing the accelerator

This example is just giving an analogy for understanding the programming steps.

The last two steps feeding and testing can by experienced by driving the motorcycle.

An efficient algorithm should have the following characteristics:

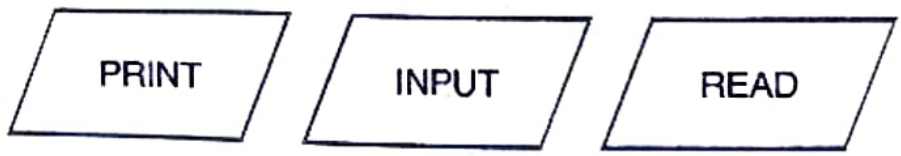
1) It should be easy and simple in language.
2) It should not repeat the task again and again.
3) Every step must be accurate and complete in itself.
4) It can be easily modified if necessary.
5) It should be easily understandable to others and steps should be clear and concise.
6) It should be economical in the use of computer time, computer storage and peripherals.
7) It should be correct for clearly defined situations.

b) Flow Charts

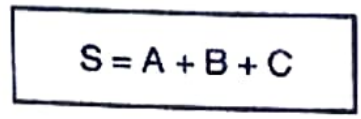
After preparing an algorithm for given task it can be presented in a pictorial form in brief. The method used to represent an algorithm in pictorial form with standard meaningful symbols is known as a **flow chart.** Flow chart is basically prepared in formulating and understanding algorithms. With flow charts programmer illustrates different loops, logical decisions, input, and output data. To draw flow charts the following figures are used as standard convention. Flow chart explains actual flow or steps in which program executes.

1) Rectangular with rounded corners to specify START and STOP known as terminal. It is usually used at Start and End of program.

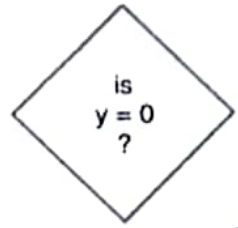
2) A parallelogram to represent input and output operation. It is known as input/output symbol. It indicates input process such as keyboard entry and output process such as printing.



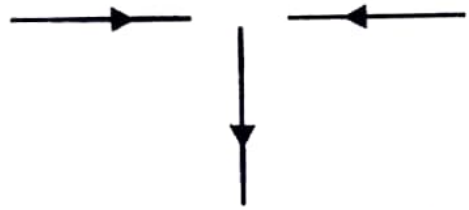
3) A rectangle is used to represent processing or calculating part.



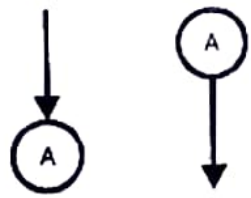
4) A diamond symbol is used to present decision-making step in the form of question and logical answer, it is known as "Decision box".



5) Arrows are used to indicate the direction of flow of program process.



6) A small circle is used to connect incomplete flow chart to its remaining part. It is known as connector. Normally it is used at the end of page where flow chart breaks and situations where flow lines cross each other.



Following rules must be considered while drawing a flow chart:

1) The standard symbols should be used.
2) The arrows should be used to represent the direction of flow.
3) Flow lines should not cross each other.
4) Program flow should be top to bottom.

Flow chart analysis gives pictorial structure of a program solution. It helps programmer in following respect: It duplicates the logic of program more easily in picture form. So it's easy for programmer to explain program logic to other using flowchart. It supports for program documentation.

It is very helpful in detecting, locating and removing mistakes in a program.

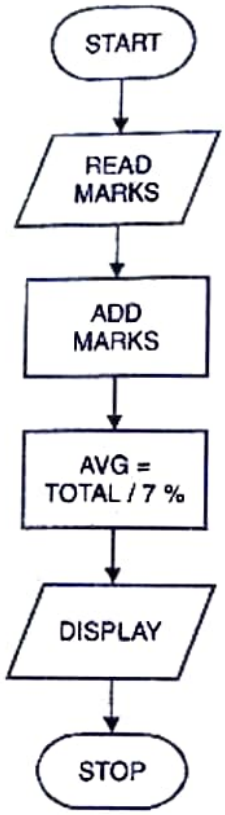
It is like a guideline from starting point of program to last point.

Advantages of Flow chart:

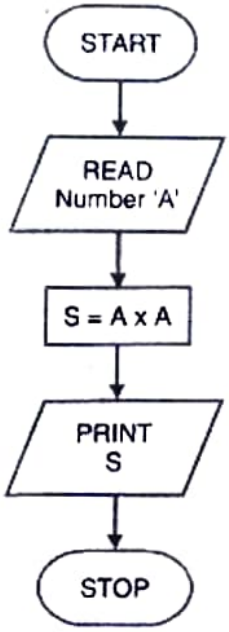
1) Better programming steps.
2) Minimum length.
3) Systematic analysis.
4) Minimization of errors.
5) Effective testing and results.
6) Easy coding.
7) Universal logical solution.

Examples:

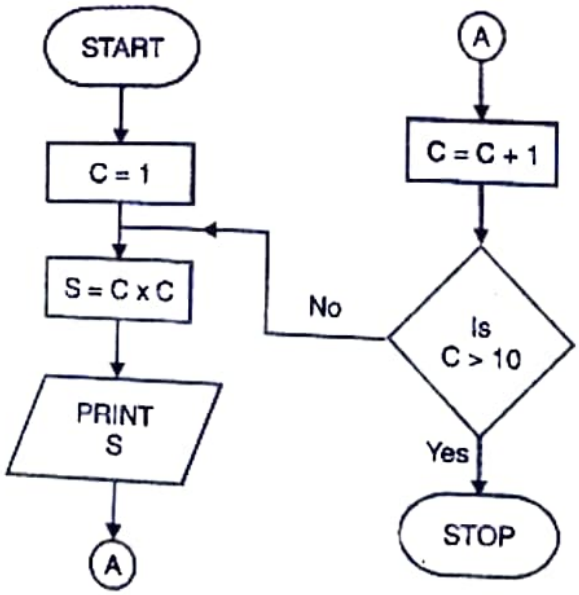
1) Draw a flow chart to find average of marks in % for a students of SSC exam.



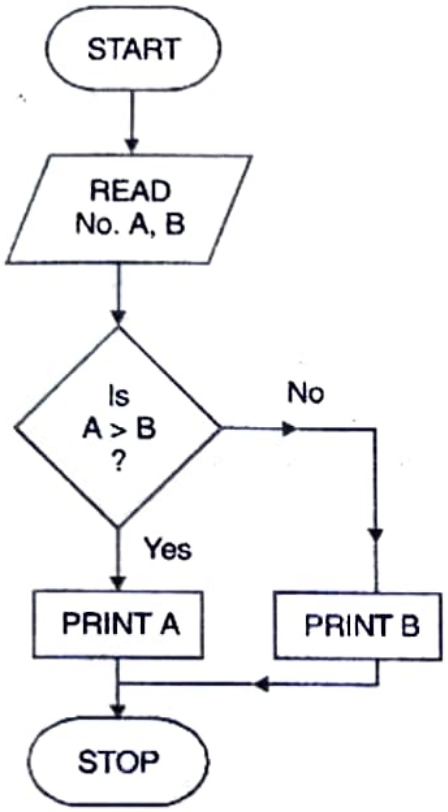
2) Flow chart to find square of given number.



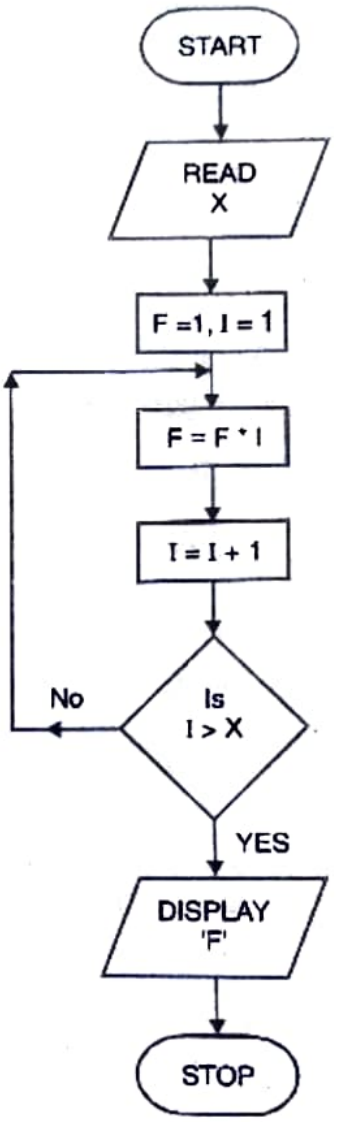
2) Flow chart to find square of numbers from 1 to 10.



4) To find largest number from given two numbers.



5) Flow chart to find factorial of 'X'.

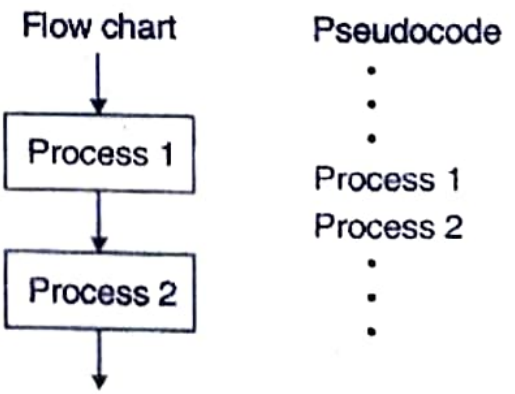


c) Pseudocode

It is a programming analysis tool used for planning program logic. Pseudo means false and code refers to the instruction written in programming language. It is an imitation of actual computer instructions. Pseudo instructions are phrases written in ordinary natural language. (e.g. English, French, German)

Pseudocode is made up of following basic logic structures that have been proved to be sufficient for writing any computer program.

1) Sequence
2) Selection
3) Iteration e.g.



Advantages of psedocode

1) Converting a pseudocode to programming language is much more easier as compared to flowchart.
2) It's easy to write pseudocode as compared to drawing flowchart (flowchart requires more time and efforts) and actual language program because pseudocode has few rules to follow. So it will concentrate on logic of program.

Limitations of psedocode

1) Graphic representation of program logic is not available in pseudocode.
2) As there are no standard rules for writing pseudocode, different programmers may use their own style, which may cause communication problem.

3.3 PROGRAMMING TECHNIQUES

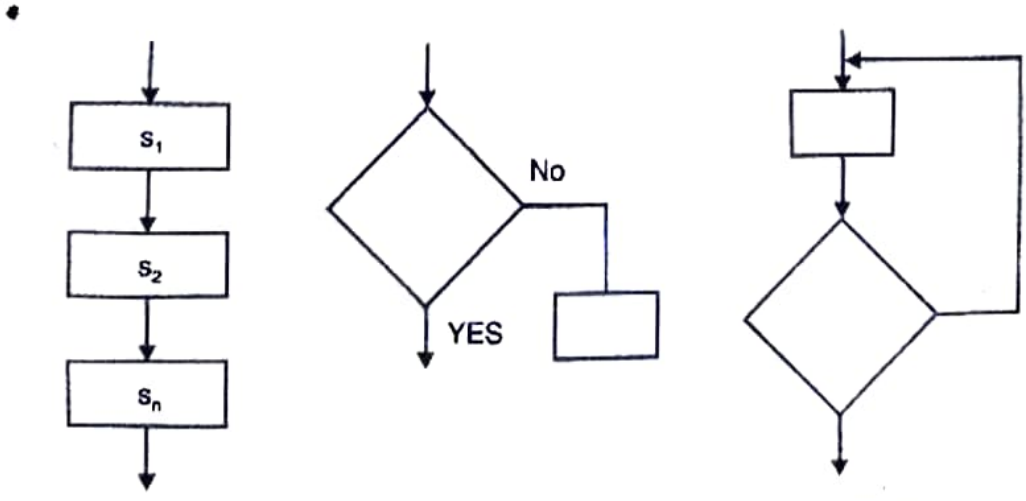
Before programming user has to decide the method in which program is to be developed. There are two basic techniques.

1) Structured programming
2) Modular programming

1) Structured Programming

In this programming method, the program is developed in hierarchy of modules where module refers to subprogram in the main program. It is a top to bottom approach. Figure shows a typical structured program. In this technique three basic structures are normally adopted such as:

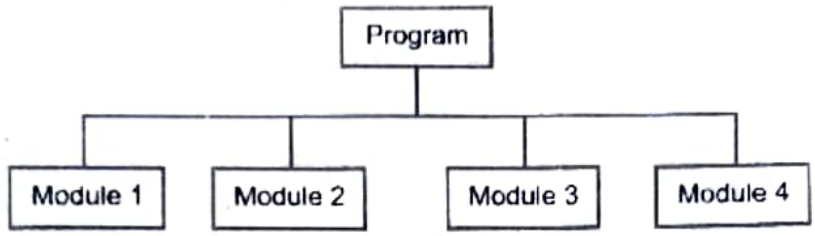
1) A simple sequence
2) A selection by decision
3) A repetition



Simple sequence Selection sequence Repetition

2) Modular Programming

In this programming method, a main module is prepared just by writing instructions of control. Submodules are prepared by writing required programs in the main program. Each subprogram may be called as "Functions" or "Procedures". Each function performs a specific routine, which may be required many times in the main program.



Modular programming reduces rewriting the subprograms. In "C" language modular programming is more popular. Modular programming allows complex problems to be divided into simple parts. All these parts (Modules) are controlled by a single program. Each module may be called as subroutine.

2.4 SEARCHING

To find any record (may be data item) matching the specific criteria in a group of items or list is called searching.

For searching various algorithms are available:

1) Linear searching
2) Binary searching

Linear Searching

If 'N' items are given and you want to find whether item is present in group of items or not, then algorithm for that is as follows:

1) Start
2) Read N and item to be found
3) Initialize counter i to 1.
4) If B[i] = item then print item found at position i.
   else increment i
5) If i < N then go to step 4.
6) If i > N then display "Item not found"
7) Stop

Here i = counter
item = element to be searched
B → array name (list name)
N → Number of items in list

Here the item to be found is compared with each element in the list.

Binary Searching

It is a simple method to search a sorted list. We assume that elements are arranged in ascending order. Procedure is as follows,

1) Calculate position of mid point element.
2) Compare value of mid point element with the number. If desired value is less than the value of mid point element then reduce the search in second half considering mid point of the interval.
3) Repeat the search if interval is not empty.
4) On successful search, position is printed.

In order to narrow down search, binary search is used.

While using this searching method array should be sorted in increasing numerical order.

Suppose DATA is an array, which is sorted in increasing numerical order, and we want to find out the location LOC of a given ITEM of information in DATA.

Then binary search algorithm is applied, it works as follows. During each search for ITEM is reduced to a segment of elements of data:

DATA [BEG], DATA [BEG +1], DATA [BEG+2],..... DATA [END]

Here BEG and END variables denote beginning and end locations of the segment under consideration. This algorithm compares ITEM with middle element DATA [MID] of the segment where MID is obtained by

MID = INT [(BEG + END)/2]

If DATA [MID] = ITEM then the search is successful and we set LOC: = MID otherwise a new segment of DATA is obtained as follows,

a) If ITEM<DATA [MID] = ITEM then ITEM can appear in left half of the segment.

DATA [BEG], DATA [BEG +1],..... DATA [MID-1]

So we reset END:=MID-1 and begin searching again

b) If ITEM > DATA [MID] then ITEM can appear only in the right half of the segment.

DATA [MID+1], DATA [MID+2],..... DATA [END]

So we reset BEG: = MID+1 and begin searching again.

c) If ITEM is not in DATA then eventually we obtain END < BEG

It means that search is unsuccessful. In this case we assign LOC: = NULL

(Generally we begin with BEG=1 and END=N or more generally we can say BIG=LB and END=UB)

Algorithm for Binary Search

(Binary Search) BINARY [DATA, LB, UB, ITEM, LOC]

Here DATA is sorted array with lower bound LB and upper bound UB and ITEM is given item of information. The variables BEG, END and MID denote respectively the beginning, end and middle location of a segment of elements of DATA. This algorithm finds the location LOC of ITEM in DATA or set LOC=NULL.

1. [Initialize segment variables.]

Set BEG: = LB, END: = UB and MID=INT [(BEG+END)/2]

2. Repeat steps 3 and 4 while BEG <= END and DATA [MID] ≠ ITEM
3. IF ITEM < DATA [MID], then:

Set END: = MID-1

Else:

Set BEG: = MID+1

[End of IF structure]

Set MID: =INT [(BEG+END)/2]

[End of step 2 loop]

IF data [MID] = ITEM, then:

Set LOC: =MID

Else:

Set LOC: =NULL

[End of IF structure]

4. Exit

Complexity of Binary Search Algorithm

The complexity is measured by the number of f(n) of comparisons to locate ITEM in DATA where DATA contains n elements. Each comparison reduces sample size in half. So we require at most f(n) comparisons to locate ITEM where

$f(n) = [\log_2 n]+1$

Applications:

(1) This algorithm is used to search ordered array.

(2) To find out the target element whether it is present or not. If present; then correspondingly give position in the array.

Limitations:

(a) The given list must be sorted.

(b) The access of list must be random means; the middle element can be accessed.

2.5 SORTING

Sorting is a process of arranging the data items in ascending or descending order. To arrange the elements in ascending or descending order in array or list or for any data structure various sorting algorithms are used.

1) Sorting by selection 2) Bubble sorting 3) Shell sorting 4) Heap sorting

Bubble sorting

Suppose the list of numbers A[1], A[2], ..., A[N] in memory.

Step 1 – compare A[1] and A[2] and arrange them in desired order so that A[1] < A[2]. Continue this comparison until we compare A[N-1] with A[N] and arrange them so that A[N-1] < A[N].

Step 1 involves n-1 comparison. So largest element is bubbled up to n^{th} position.

Step 2 – repeat step 1 with one less comparison. Now we stop after we compare A[N-2] and A[N-1], so second largest element is bubbled at A[N-1]

: :

Compare A[1] with A[2] and arrange them so that, A[1] < A[2]. After N-1 steps the list will be sorted in increasing order.

SOLVED EXAMPLES

1) Write an algorithm to find whether given number is prime or not.

Prime number is positive integer, which is divisible by itself and 1.

Algorithm:

1) Input number n.
2) If n < = 1, number is not prime and stop.
3) If n = 2, it is prime and stop.
4) Let I = 2
5) Repeat
   a) Divide n by I, if remainder is 0 then n isn't prime and stop.
   b) I= I + 1 until I < = N – 1.
6) Stop.

2) Write an algorithm to find factorial of given number

Algorithm:

1) Input n.
2) Set product = 1 and count to 1
   While count < = N repeat.
   a) Product = product * count
   b) Count = count + 1.
3) Product is actual factorial.
4) Stop.

3) Write an algorithm to find Fibonacci Sequence

0, 1, 1, 2, 3, 5, 8, 13

In Fibonacci sequence the first two items are fixed i.e. 0 and 1. After that each term is sum of two nearest predecessor terms.

1^{st} term → i.e. a = 0

2^{nd} term → i.e. b = 1

the new term C = a + b

again change a and b

a = b

b = C

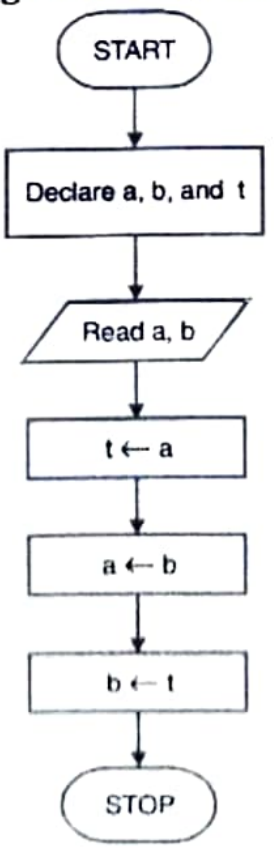
and again new term C = a + b

This continues until the n^{th} term.

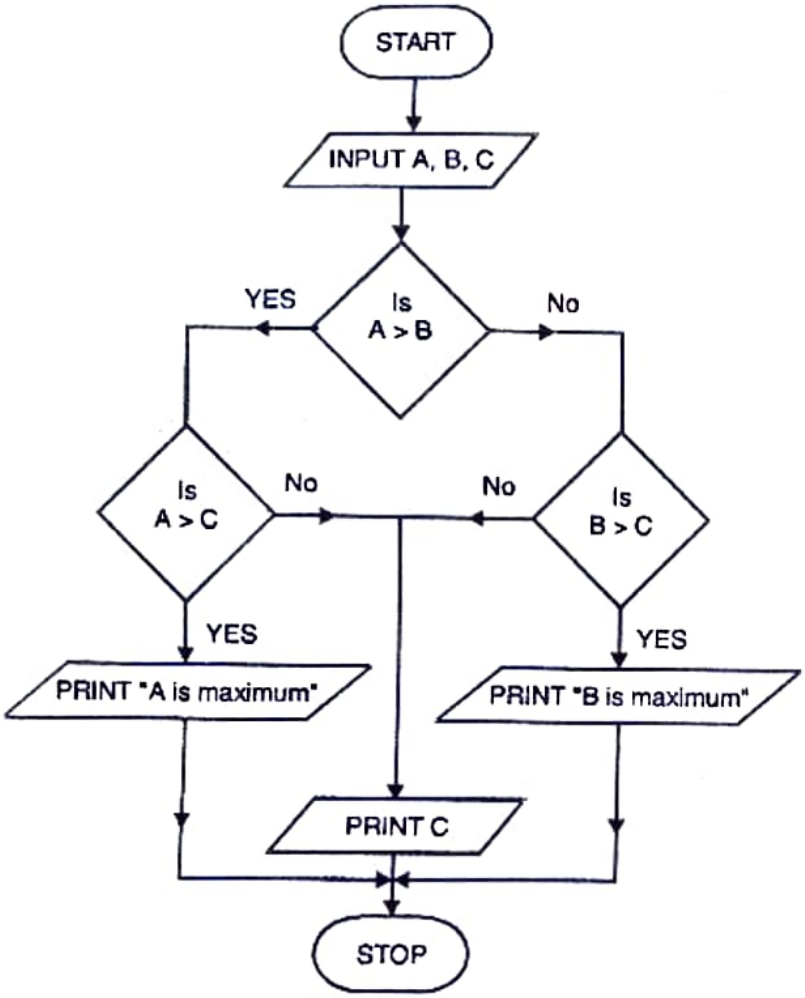
Algorithm:

1) Read n. the number of fibonacci terms to be generated.
2) 1^{st} term a = 0, 2^{nd} term b = 1
3) Initialize counter = 2
4) While counter C <= n repeatedly do
   a) C = a + b and display C
   b) a = b, b = C
5) Stop

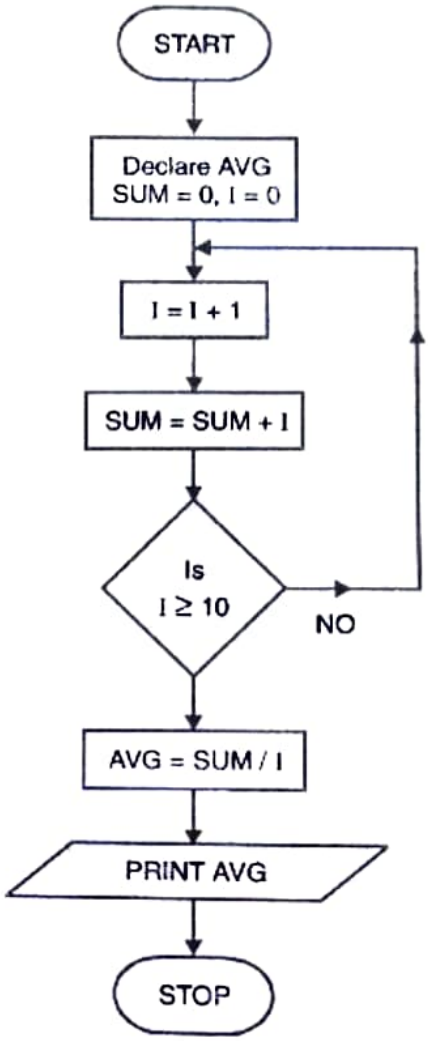
3) Draw a Flowchart to exchange values of two variables



1) Draw a Flowchart to find maximum among given 3 numbers.
Number given – A, B, C



5) Draw flow chart to find number average of first 10 numbers.



QUESTIONS

1) Select the correct alternative and rewrite the following:

1) A diamond box is used to indicate _____ in flow charts.
   a) Input, b) Output, c) decision, d) none of these

2) Writing the program task in simple sequence and in simple English is known as _____.
   a) algorithm, b) flow chart, c) code, d) none of these

3) Arranging the items in list in descending or ascending order is called _____.
   a) searching, b) sorting, c) finding, d) none of these

4) _____ pictorial representation of a logic of a program.
a) flow chart, b) chart, c) pseudocode, d) picture code

5) The number of comparison in linear search are _____ that of binary search.
a) less than, b) greater than, c) equal to, d) none of these.

6) _____ is midway between algorithm and program.
a) flowchart, b) pseudocode, c) source code, d) none of these

Aneswer: 1) decision 2) algorithm 3) sorting 4) flowchart
5) greater than 6) pseudocode

2. What ate the steps involved in program development? Explain in brief.
3. What are various symbols used while drawing flow chart? Explain it.
4. Define and Explain algorithm concept with suitable example.
5. What are the characteristics of algorithm?
6. Define Flow chart and write the advantages of flow charts?
7. Explain following terms:

   a)Sorting, b) Psedocode
8. What is difference between modular programming and structured programming?
9. What is sorting? State various algorithms used for it.
10. What is binary searching? Explain it.
11. Explain whole programming process in brief.
12. Write algorithm and draw flow charts for the following problems:

    b) To find minimum number among given 3 numbers.

    c) To check whether the number is odd or even.

    d) To calculate sum of all odd numbers between 1 and 100.

❑❑❑

3. INTRODUCTION TO C++

INTRODUCTION

Bjarne Stroustrup at Bell Labs initially developed C++ during the early 1980's. It was designed to support the features of C such as efficiency and low-level support for system level coding. Added to this were features such as classes with inheritance and virtual functions, derived from the Simula67 language, and operator overloading, derived from Algol68. C++ is best described as a superset of C, with full support for object-oriented programming. This language is in wide spread use.

C++ programs are fast and efficient, qualities, which helped make C++ an extremely popular programming language.

Getting started:

To write any C++ program (usually known as source code and file in which you save this program is known as source file.),you need a text editor.**Text editor** is software(supported by underlying platform) that allows you to create and edit text files.

While after coding is over, to run your program, that should be made error free, this process is known as compilation and compiler is a software tool which is doing this job. It converts your source code into machine understandable code(known as object code).It also displays the grammatical(syntactical)errors in your program.

C++ programs can be created using any text editor. e.g. on Unix Operating System platform, use vi or ed text editor for creating and editing the source code. While on DOS Operating System, we can use edlin or any other editor available.

Some systems like **Turbo C++** and **Borland C++** provide an integrated program development environment **(IDE)** under MS-DOS. They provide a built-in editor and a menu bar, which includes options such as file, edit, compile and run. We can create and save the source files under the file option and edit them under the edit option. (C++ programs are saved using file extension **.cpp**). We can then compile the program under the compile option and execute it under the run option. The run option can be used without compiling the source code. In this case, the run command causes the system to compile, link and run the program in one step.

Another programming environment by Microsoft that runs under Windows is **VISUAL C++(VC++).** Visual C++ is a visual programming environment in which basic program components can be selected through menu choices, buttons, icons and other predetermined methods. (Refer practical section for TCC and VC++ editors.)

Writing a Simple C++ program:

```
1. #include <iostream.h>
2. void main ()
3. {
4. cout << "Hello and Welcome to world of C++\n";
5. }
```

Output is :-- Hello and Welcome to world of C++.

Explanation of the program:

Here the first line in the program deals with the including of header file (iostream is Input –Output stream and known as preprocessor directive).It is needed to support all the in built io functions like cout (in this example.).The identifier **iostream** is the name of the file in standard C++ library. iostream included in <> brackets is known as standard header.

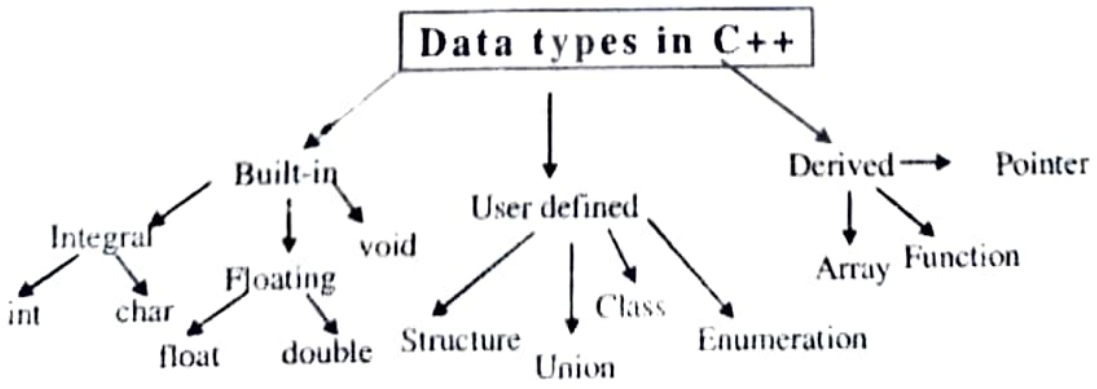
Second line is the main function within the body of which you start writing the code for your program. Every C++ program must include one and only one main function.

3rd and 5 th line { and } actually contains program body.4th statement will output the given message on screen. Symbol << represents C++ output operator. When this statement gets executed, the characters enclosed in quotation marks (" ") are sent to standard output device(screen).

Now let's study various programming elements required to write a C++ program.

3.1 DATA TYPES IN C++

The following chart shows different data types in C++.



Built-in data types

Name	Bytes*	Description	Range*
char	1	Character or integer 8 bits length.	signed: -128 to 127 unsigned: 0 to 255
short	2	Integer 16 bits length.	signed: -32768 to 32767 unsigned: 0 to 65535
long	4	Integer 32 bits length.	signed:-2147483648 to 2147483647 unsigned: 0 to 4294967295
int	*	Integer. Its length traditionally depends on the length of the system's Word type, thus in MSDOS it is 16 bits long, whereas in 32 bit systems (like Windows 9x/2000/NT and systems that work under protected mode in x86 systems) it is 32 bits long (4 bytes).	See short, long
float	4	floating point number.	3.4e + / - 38 (7 digits)
double	8	double precision floating point number.	1.7e + / - 308 (15 digits)
long double	10	long double precision floating point number.	1.2e + / - 4932 (19 digits)
bool	1	Boolean value. It can take one of two values: true or false NOTE: this is a type recently added by the ANSI-C++ standard. Not all compilers support it.	true or false

Values of columns Bytes and Range may vary depending on your system. The values included here are the most commonly accepted and used by almost all compilers.

Both C and C++ compilers support all the built-in (also known as basic or fundamental) data types. Here **void** is a generic "nontype." void type is used as the return type for functions that do not return a value. With the exception of void, the basic data types may have several modifiers preceding them to serve the needs of various situations.

The modifiers signed, unsigned, long and short may be applied to character and integer basic data types. However, the modifier long may also be applied to double. Data type representation is machine specific in C++.

Type Modifiers

unsigned :--Doesn't use sign bit (assumes int if base type is omitted).
long :--May have twice as many bytes as base type (assumes int if base type is omitted).
short:-- May have half as many bytes as base type (assumes int if base type is omitted).
const :--Constant (values can't be changed during execution).

- **Compound types -**

a) pointer
b) arrays
c) functions
d) references
e) classes
f) unions
g) enumerations

(we will study these in the next Part.)

3.2 VARIABLES

Declaration of a variable involved name of the variable & its data type.Variables can be declared anywhere before they are used ..

Syntax: data-type var 1, var 2, ... ;

e.g.: int varname ;

and they are optionally initialized by the assignment of a literal value

e.g. float varname= 0 ;

Multiple variables of the **same type** are declared as a **comma-delimited** list after the type, e.g. char var1,var2;

A valid variable name may consists of alphabet, digits, underscore, but it can't start with a digit.

e.g. int 45abc ; is not a valid variable name since it starts with digit.
float f=3.14 23 ; is not a valid variable name since it contains space.

(Refer topic **identifiers**)

In C++ uppercase and lowercase variables are treated as different variables.

Variables have scope, that defines where from they can be refereed. Accordingly they can have global and local scope. If declared outside functions, variables have **global scope** (i.e. can be referenced from anywhere in the program)

While if declared inside a function, variables have **local scope** - i.e. can be referenced only from within that function. (for details refer topic **scope and visibility**.)

3.3 CONSTANTS

Constants: are declared with the "**const**" keyword, a type, their name, and an assignment of their constant value. This value must match the declared type, and is **fixed** for the duration of the program.

Syntax: const data type const_name = value;
e.g. const float Pi = 3.14;
const char Blank = ' ';

Types of Constants

1) Numerical constants

a) integer constant –declared with const keyword and has fixed integer value.
e.g. **const** int i = 10;

b) character constant : declared with const keyword and has fixed character value.
e.g. **const** char a = '\t';

When C++ compiler encounters such a character constant, it translates it into the corresponding ASCII code and stores that number (ASCII character set is a way of representing characters).

Character escape sequences: Back slash (\) alters the meaning of the character that follows it.

e.g. the character 'a' when typed after backslash, i.e. **\a** will mean to produce an available or visual alert signal.

Backslash character	Meaning
\a (alert)	Produces an available or visible alert signal.
\b (backslash)	Moves the cursor bar one space
\f (form feed)	Moves ..... to the next page
\n (new line)	Prints a new line
\r (carriage return)	Prints a carriage return
\t (horizontal tab)	Prints a horizontal tab
\v (vertical tab)	Prints a vertical tab

c) Floating point constant: Also called as real constants.
e.g. **const** double pi = 3.14;
it can also be written using exponential notation.
e.g. number 1234.56 is written as 1.23456E3. It indicates how many places the decimal point must be moved to change the number to the ordinary decimal notation.

Suffixes for Numerical Constants

Integer constants default to the smallest integer type that can hold their value. Otherwise, the following suffixes can be used (alone or together): **u for Unsigned** or **L for Long**

Floating point constants default to type *double*. Otherwise, the following suffixes can be used: **f for float** or **L for long double**

d) String constant: A sequence of two or more characters surrounded by double quotes is called a string constant. This can also be defined as an array of character constants.
e.g. char arr[5]="abcde" ;
(refer topic arrays in the next part.)

e) Hexadecimal and octal constants: In day to day use, we use decimal constants having base 10 .Numbers can also be specified using base 8 & 16. Numbers represented using base 8 are having octal number system (uses 0 – 7 digits) while having base 16, are having Hexadecimal number system (uses numbers from 0 to 9 and letters A to F)
The integer constants can be specified in hexadecimal or octal form.
The octal number begins with letter 0 and hexadecimal number begins with 0x.
e.g. 0xAB is a hexadecimal no.
0156 is a octal number

2) Defined constants
#define Directive(PREPROCESSOR directive:)
The #define directive, causes a macro or symbolic name to be defined as a macro.ie.it sets up an equivalence between identifier and a text phrase.
Syntax: #define macro_name replacement_list
Or #define identifier text phrase
e.g. #define PI 3.14159 appearing at the beginning of the program specifies that the identifier PI will be replaced by text 3.14159 throughout the program. No semicolon should be specified. It can appear anywhere in the program, but normally specified at the beginning of your main program.

e.g #define NEWLINE '\n'
#define arrSize 100 so the line char arr[arrSize] expands as char arr[100];

3) Symbolic constants: -- Enumerations
In addition to the predefined data types C++ allows you to define your own special data types. This is done using several ways; one of the ways is to use Enumeration. An enumeration of symbolic constants is introduced by an **enum** declaration. This is useful for declaring a set of closely related constants.

Syntax:-- enum typename {enumeration –list };
Here **enum** is a keyword, typename defines identifier that names the type being defined, and enumeration list stands for a list of names of the integer constants.

e.g enum {north, south, east, west}; introduces four enumerators which have integral values starting from 0 (i.e., north is 0, south is 1, etc.)
The default numbering of enumerators can be overruled by explicit initialization:
like, enum {north = 10, south, east = 0, west}; Here, south is 11 and west is 1.
An enumeration can also be named, where the name becomes a user-defined type. This is useful for defining variables that can only be assigned a limited set of values.

e.g
```
enum Days {Sunday, Monday, Tuesday, Wednesday , Thursday, Friday, Saturday};
Days d;
```
So here d can only be assigned one of the enumerators for Days.
Use: Enumerations are particularly useful for naming the cases of a switch statement. (Refer switch-case control structure)

e.g.
```
switch(d)
  {
    case Sunday:      //...
    case Monday:      //...
    case Tuesday:     //...
    case Wednesday:   //...  etc…
  }
```

3.4 COMPILER TOKENS

When a C++ compiler analyses a program, it breaks the program into tokens. A *token* is a distinct unit that is recognizable by the compiler. There are five categories of tokens:

- keywords
- literals
- identifiers
- operators and
- white space includes: blanks, horizontal and vertical tabs, new line and form feed characters and comments

Notes:

In order to separate keywords, literals and identifiers, some white space is required within a program. White space may also be used to provide documentation in the form of comments (comments are explained later in this chapter) and to make the program readable (e.g. by indenting statements). A C++ compiler takes each token to be the longest sequence of characters that can be used to form that token. Of course, it does this by starting at the beginning of the program and proceeding in a forward direction through the source code.

Now let's study each of the tokens one by one.

KEYWORDS

Following are the keywords of the C++ language. (total 48).Keywords should be considered to be reserved. Keywords are not allowed to be used as variable names. They are as follows:

asm ,auto ,break ,case ,catch ,char ,class, const, continue ,default ,delete, do ,double ,else enum, extern ,float ,for ,friend ,goto ,if ,inline, int ,long, new ,operator, private, protected, public, register,return, short, signed, sizeof, static, struct, switch, template, throw, try, typedef, union, unsigned, virtual, void, volatile, while

INDENTIFIERS

- A valid identifier is a sequence of one or more letters, digits or underline symbols (_).
- there is no limit to the length of an identifier, , although for some compilers only first 32characters of an identifier are significant
- Neither spaces nor marked letters can be part of an identifier. Only letters, digits and underline characters are valid
- the underscore character '_' counts as a letter, In addition, variable identifiers should always begin with a letter. They can also begin with an underline character (_), In no case they can begin with a digit.
- case is significant and all characters are considered significant.
- **language's keywords** should not be used to define an identifier

LITERALS:

Literal may be, character constant, integer constant, floating point constant, string constant.

3.5 OPERATORS IN C++

An operator does some operation on operands and this operation is normally specified by operator's symbol. C++ has many operators, mainly classified as **unary** operators, working on single operand and **binary**, working on 2 operands.

Although C++ allows us to multiple meanings to the operators, yet their association and precedence remains same. E.g. (*) multiplication operator is having higher precedence than (+) add operator.

Following table lists all the C++ operators in order of non-increasing precedence. An expression involving operators of equal precedence is evaluated according to the associability of the operators.

Operator(s)	Description(s)	Associativity
::	Global scope (unary)	right to left
()	Function call	left to right
()	Value construction	left to right
[]	Array element reference	left to right
->	Pointer to class member reference	left to right
.	Class member reference	left to right
-, +	Unary minus and plus	right to left
>++, --	Increment and decrement	right to left
!, ~	Logical negation and 1's complement	right to left
*, &	Pointer dereference (indirection) and address	right to left
Sizeof	Size of an object	right to left
(*type*)	Type cast (coercion)	right to left
>*	Pointer to member selector	left to right
*	Pointer to member selector	left to right
*, /, %	Multiplication, division and modulus (remainder)	left to right
+, -	Addition and subtraction	left to right
<<, >>	Shift left and shift right	left to right
<, <=, >, >=	Less than, less than or equal, greater than, greater than or equal	left to right
==, !=	Equality and inequality	left to right
&	Bit wise AND	left to right
^	Bit wise XOR	left to right
|	Bit wise OR	left to right
&&	Logical AND	left to right
||	Logical OR	left to right

? :	Conditional _expression	right to left
=, *=, /=, %=, +=, =, &=, ^=, |=, >>=, <<=	Assignment	right to left
,	Comma	left to right

Brackets are evaluated from the inside out, when dealing with multiple operators of the same level. Level evaluate from left to right

Some of the important operators

1) **Arithmetic operators:** They are used for mathematical calculations. An arithmetic expression is made up of constants, variables, a combination of both or a function call, connected by arithmetic operators. Following are the subtypes:

a. **Unary arithmetic operator**
Needs only one operand.
e.g. +a, -b uses unary operator namely + (positive) and – (negative).

b. **Binary arithmetic operator**
Needs 2 operands.
Following arithmetic separators need 2 operands.
+ (Addition or plus)
- (Subtraction or minus)
* (Multiplication or star)
/ (Slash or division)
% (Modulus separators for remainder)
This operator can't be applied with floats or double type.
e.g. a + b, c / d, a % b, etc.
Here it gives remainder after division.
e.g. 5 % 2 will give 1 as a remainder while 5 / 2 will give 2 as a quotient.

c. **Ternary arithmetic operator**
Needs 3 operands. This is one of the special features of C++ language. (See **conditional** operator later in this chapter).

2) Increment and decrement operators
In some situations, a need may arise to increase or decrease, value of a variable by 1. C++ provides special operators ++ and – – to do the same.
Mind it, these operators are unary i.e. warning on a single operand.

Also it is illegal to increment or decrement constant or floating-point data type.
The incrementing and decrementing can be of 2 types:

a. **Prefix:** first increment/decrement the value of the variable and then take this new value for processing. e.g. ++a, --b
b. **Postfix:** First take the value of the variable increment or decrement the variable. E.g. a++, b—

Use of Prefix and Postfix

Suppose initial values of a and b are 5 and 5 respectively.
Suppose, I do postfix operation on it i.e. a++ and b—and print them. Here values of a and b remain unaltered. i.e. 5 and 5 only.
While after postfix operation is done, values of a and b are printed them will be changed to 6 and 4 respectively.

3) Relational operator:

These are used to test the relation between 2 values. All C++ operators are binary operators and hence require 2 operands.
A relational expression is made up of 2 arithmetic expressions connected by a relational operator. It returns zero when the relation is false while nonzero when it is true.

Operator	Symbol	Form	Result
Less than	<	a < b	Returns 1 if a is less than b otherwise 0
Greater than	>	a > b	Returns 1 if a is greater than b otherwise 0
Less than Or equal to	<=	a <= b	Returns 1 if a is less than or equal to b otherwise 0
Greater than Or equal to	>=	a >= b	Returns 1 if a is greater than or equal to b otherwise 0
Equal to equal to	= =	a = = b	Returns 1 if both are equal (same)
Not equal to	! =	a ! = b	Returns 1 if both values are not equal

These operators are having lower precedence than arithmetic operators.
e.g. a + b * c > d % f is evaluated as
(a + (b * c)) > (d % f)

4) Logical operators: (Boolean operators)

They combine the results of one or more expression and the resultant expression is called as logical expression.

After testing the conditions, they return logical status (true or false)
They may be unary or binary operators, and the operands may be constants, variables or even expressions. The operands may be integers/floating point numbers.

Operator	Symbol	Form	Result
Logical AND	& &	a & & b	Returns 1if a and b are nonzero else returns 0
Logical OR	||	a || b	Returns 1 if a or b is nonzero, else 0
Logical negation	!	! a	Returns 1 if a is zero else returns 0

e.g.
```
if ((a < b) && (a < c))
{
        cout << "a is smallest";
}
```

Hierarchy of logical operators --
Logical NOT (!) operator has a higher precedence than other.
While AND (&&) has higher precedence than logical OR (||)
They have lower precedence than the relational and arithmetic operators.
Truth (Boolean) table for **! operator** – takes only 1 operand(used for negation)

Operator	Operand	Result
!	(false) 0	1
!	(true) 1 or nonzero	0

Truth (Boolean) table for **&& operator:** Takes 2 operands.

Operand 1	Operator (&&)	Operand 2	Result
False (0)	&&	False (0)	0
True (nonzero)	&&	False (0)	0
False(0)	&&	True (nonzero)	0
True (nonzero)	&&	True (nonzero)	1 (nonzero)

If both the operands are true (nonzero) or if they are the expression returning true value then only result is true otherwise false.

Truth (Boolean) table for || **operator:** Takes 2 operands.

Operand 1	Operand (||)	Operand 2	Result
False (0)	||	False (0)	0
True (nonzero)	||	False (0)	1
False(0)	||	True (nonzero)	1
True (nonzero)	||	True (nonzero)	1 (nonzero)

Returns false (0) only when both the operands or expressions returns false otherwise always true.

5) Conditional operator (?:)

It is **ternary** operator taking 3 arguments. The value at an _expression using conditional operator is the value of either its.2nd or 3rd operand, depending on the value of the first operand.

This is equivalent to if.else control structure. (Refer if...else control structure in the next part)

Operator	Symbol	Form	Result
Conditional	? :	a ? b : c	If a is nonzero, result is b, otherwise it is c

In the above statement, the first operand is the test condition while 2nd and 3rd represents the final value of the expression. The following expression assigns greater value of a & b to variable Max.

Max = ((a > b) ?a : b) ;

6) Assignment operator (=)

= operator causes the value of the right hand operand to be assigned to the left hand operand.

The left hand operand, sometimes called as lvalue, must always refer to a memory location.

This operator has **right-to-left** associativity.

e.g. x = y = z = 5: is interpreted as, (x = y = (z = 5));

7) Shorthand (compound) operators

C++ supports additional assignment operators combining assignment with each of the arithmetic operations.

Operator	Symbol	Form	Operation
Add-assign	+ =	a + = b	Equivalent to a = a + b
Subtract-assign	- =	a - = b	Equivalent to a = a - b
Multiply-assign	* =	a * = b	Equivalent to a = a * b
divide-assign	/ =	a / = b	Equivalent to a = a / b
Remainder-assign	% =	a % = b	Equivalent to a = a % b

8) Bitwise operators:

Some special C++ operators can also work on the bits, so they may be called to as bit manipulation operators.
They are used to manipulate or modify the individual bits of the piece of data.
They can be used only with the integral built-in data type i.e. in connection with char or int.

Operator	Form	Meaning
~	~ a	One's complement
>>	a >> b	Right shift a by b
<<	a << b	Left shift a by b
&	a & b	Bitwise AND
|	a | b	Bitwise OR
^	a ^ b	Bitwise XOR

a) One's complement operator (~)

The number is complemented (1's complement) by replacing all 1's by 0's and vice versa. e.g. 1's complement of 1100 is 0011.

b) Right shift operator (>>)

Operator on the single operand.
Shifts each bit in the operand to the right.
Number of places the bits are shifted, depends on the number following the operand.
e.g. Suppose binary representation of a number x is 1001 1100
then x >> 1 gives 0100 1110
When number is shifted to right, left place is emptied and filled with 0.

Shifting operand one bit to the right is equivalent to dividing that number by 2.
e.g. 32 >> 1 gives 16
32 >> 2 gives 8

c) Left shift (<<) operator
It shifts each bit in the operand to the left and zeros are added to right of the number.
e.g. binary representation of a number is 11001001
So when it is left shifted by 1, it's equivalent is 1001 0010
Left shifting by 1 is equivalent to multiplying number by 2.

d) Bitwise AND operator (&)
It works on 2 operands and both the operands should be the same data type as well as either char or int.
The purpose of AND operator (&) is to clear the given bit.

1st bit	2nd bit	1st bit & 2nd bit
0	0	0
0	1	0
1	0	0
1	1	1

e.g. 1100 0111
&
1011 0100
Result 1000 0100

e) Bitwise OR operator (|)
It also works on 2 operands. It is used to set the resultant bit to 1.

1st bit	2nd bit	1st bit | 2nd bit
0	0	0
0	1	1
1	0	1
1	1	1

e.g. 1100 1001
|
1001 1101
Result 1101 1101

f) Bitwise XOR (^) operator

It works on 2 operands. It sets the resultant bit if both the bits on which it operators are not equal.

Bit 1	Bit 2	Bit 1 ^ Bit 2
0	0	0
0	1	1
1	0	1
1	1	0

e.g.
```
  1011 1000
     ^
  1100 1001
  ---------
  0111 0001
```

9) Operators dealing with streams

a) Insertion operator (<<)

Stream decides flow of data. There are many types of streams in C++,but dealing with input and output is called *iostream.*The operator dealing with output stream is called insertion or **put to** operator. It inserts the contents of the variable on its right to the object on its left. This operator is used along with **output stream.**

e.g. `cout << a;`

Here value at variable a declared in the program is printed on standard output device.(screen)

While, `cout << "a";` will display character a (included in double quotes) on the screen.

b) Extraction operator (>>)

It is **get from** operator. It extracts/takes the value from the standard input device i.e. keyboard and assigns is to the variable on its right. It is associated with the input stream.

e.g. `cin >> a;`

So value entered by the user from keyboard is assigned to variable a declared in the program.

10) Comma Operator

Multiple expressions can be combined into one expression using the comma operator. The comma operator takes two operands. It first evaluates the left operand and then the right operand, and returns the value of the latter as the final outcome.

e.g
```
int m, n, min;
int mCount = 0, nCount = 0;
```

```
//...
min = (m < n ? mCount++, m : nCount++, n);
```

Here when m is less than n, mCount++ is evaluated and the value of m is stored in min. Otherwise, nCount++ is evaluated and the value of n is stored in min.
This operator can also be used in multiple assignments in for loop
e.g for(int i=0,int j=1 ; i<=10 ; i++,j--)

11) The sizeof Operator

C++ provides a useful operator, **sizeof,** for calculating the size of any data item or type. It takes a single operand, which may be a type name (e.g., int) or an expression (e.g., 100) and returns the size of the specified entity in bytes. The outcome is totally machine-dependent. Following program illustrates the use of sizeof on the built-in types we have encountered so far.

```
#include <iostream.h>
void main ()
 {
        cout << "char   size = " << sizeof(char) << " bytes\n";
        cout << "char*  size = " << sizeof(char*) << " bytes\n";
        cout << "short  size = " << sizeof(short) << " bytes\n";
        cout << "int    size = " << sizeof(int) << " bytes\n";
        cout << "long   size = " << sizeof(long) << " bytes\n";
        cout << "float  size = " << sizeof(float) << " bytes\n";
        cout << "double size = " << sizeof(double) << " bytes\n";
        cout << "1.55   size = " << sizeof(1.35) << " bytes\n";
        cout << "1.55L  size = " << sizeof(1.45L) << " bytes\n";
        cout << "string size = " << sizeof("welcome") <<"bytes\n";
 }
```

When run, the program will produce the following output

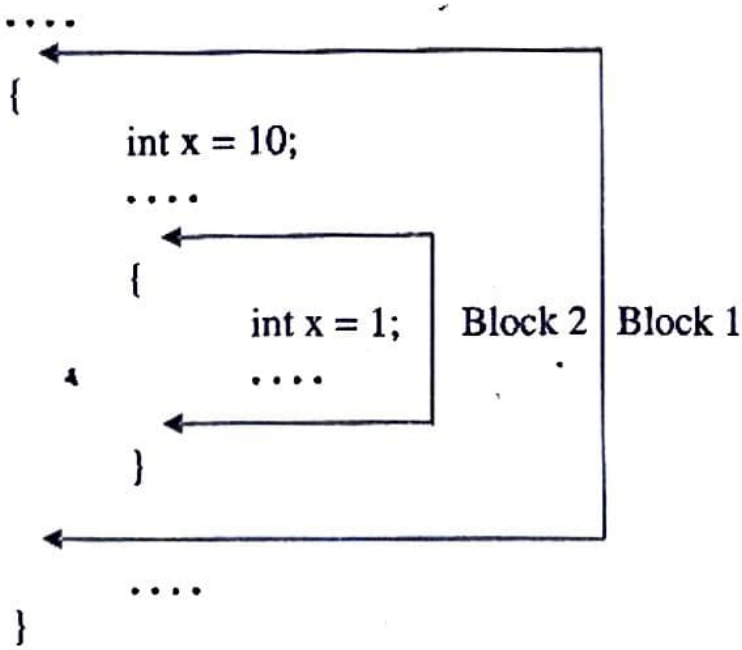
```
char size = 1 bytes
char* size = 2 bytes
short size = 2 bytes
int size = 2 bytes
long size = 4 bytes
float size = 4 bytes
double size = 8 bytes
1.35size = 8 bytes
1.45L size = 10 bytes
string size = 7 bytes
```

12) Scope Resolution Operator

C++ is also a block-structured language. Blocks and scopes can be used in constructing programs. We know that the same variable name can be used to have different meanings in different blocks. The scope of the variable extends from the point of its declaration till the end of the block containing the declaration. A variable declared inside a block is said to be local to that block. Consider the following segment of a program:

```
....
{
int x = 10;
....
}
....
{
int x = 1;
....
}
```

The two declarations of x refer to two different memory locations containing different values. Statements in the second block can not refer to variable x declared in the first block and vice versa. Blocks in C++ are often nested. For example, the following style is common:



Block 2 is contained in block 1. Note that a declaration in an inner block hides a declaration of the same variable in an outer block and therefore, each declaration of x causes it to refer to a different data object. Within the inner block, the variable x will refer to the data object declared therein.

In C, global version of a variable cannot be accessed from within the inner block. C++ resolves this problem by introducing a new operator : : called the scope resolution operator. This can be used to uncover a hidden variable. It takes the following form:

Syntax :: variable_name ;

This operator allows access to the global version of a variable. Following program illustrates difference between local and global variable

```
# include <iostream.h>
int m = 10;                          // global m
void main ( )
{
        int m = 20;                  // m redeclared, local to main
        {
                int k = m;
                int m = 30;          // m declared again
                                     // local to inner block
                cout <<"we are in inner block\n";
                cout <<"k =" <<k<<"\n";
                cout <<"local m =" <<m<<"\n";
                cout <<" global m = "<< : : m <<"\n";

        }
}
output is : we are in inner block
            k =20
            local m =30
            global m = 10
```

12) Memory Management Operators (new, delete)

We use dynamic allocation technique when it is not known in advance how much of memory space is needed. C++ supports define two unary operators **new** and **delete** that perform the task of allocating and freeing the memory in a better and easier way. Since these operators manipulate memory on the free store, they are also known as **free store operators.**

(Also refer topic **pointers**)

An object can be created by using new, and destroyed by using delete, as and when required. A data object created inside a block with new will remain in existence until using delete explicitly destroys it·

The new operator can be used to create objects of any type. It takes the following general form:

Syntax: pointer_variable = **new** data type;

Here, pointer_variable is a pointer of type data-type. The **new** operator allocates sufficient memory to hold a data object of type data-type and returns the address of the object. The data-type may be any valid data type. The pointer-variable holds the address of the memory space allocated.

e.g. p = new int ; q = new float ;

Where p is a pointer of type int and q is a pointer of type float. Here, p and q must have already been declared as pointers of appropriate types. Alternatively, we can combine the declaration of pointers and their assignments as follows:

```
int *p = new int;
float *q = new float;
```

Subsequently, the statements

```
* p = 25;
* q = 7.5;
```

will assign 25 to the newly created int object and 7.5 to the float object.

13) type cast operator.

C++ permits explicit type conversion of variables or expressions using the type cast operator.

Syntax : type-name (expression)

e.g. average = sum/float (i);

Type conversions: Simple Type Conversion

A value in any of the built-in types we have seen so far can be converted (type-cast) to any of the other types. For example:

```
(int) 3.14                // converts 3.14 to an int to give 3
(long) 3.14               // converts 3.14 to a long to give 3L
(double) 2                // converts 2 to a double to give 2.0
(char) 122                // converts 122 to a char whose code is 122
(unsigned short) 3.14     // gives 3 as an unsigned short
```

As illustreted by these examples, the built-in type identifiers can be used as type operators. Type operators are unary (i.e., take one operand) and appear inside brackets to the left of their operand. This is called explicit type conversion. When the type name is just one word, an alternate notation may be used in which the brackets appear around the operand:

int(3.14) is same as (int) 3.14

Implicit Conversions (automatic conversions)

We can mix data types in expressions

e.g m = 5 + 2.75

is a valid statement. Wherever data types are mixed in an expression, C++ performs the conversions automatically. This process is known as implicit or automatic conversions. When the compiler encounters an expression, it divides the expressions into sub-expressions consisting of one operator and one or two operands. For a binary operator, if the operands type differs, the compiler converts one of them to match with the other, using the rule that the "smaller" type is converted to the "wider" type. For example, if one of the operand is an int and the other is a float the int is converted into a float because a float is wider than an int. This happens when values of different types are mixed in an expression.

```
e.g. double d = 1;  // d receives 1.0
     int i = 10.5;  // i receives 10
     i = i + d;     // means: i = int(double(i) + d)
```

In the last example, i + d involves mismatching types, so i is first converted to double (promoted) and then added to d. The result is a double, which does not match the type of i on the left side of the assignment, so it is converted to int (demoted) before being assigned to i.

14) stream manipulation operators

For formatting the data display, manipulator operators are used. Normally used with (<<) operator to modify or manipulate the way the data is displayed. These are present in <iomanip.h>header file.

Some of the useful stream manipulators are:--

dec	Sets base 10 integers.
endl	Sends a new line character.
ends	Sends a null (end of string) character.
flush	Flushes an output stream.
hex	Sets base 16 integers.
oct	Sets base 8 integers.
setw(*int*)	Sets field width.

```
e.g.  cout <<"a="<<a<<endl;
      cout <<"b="<<b<<endl;
```

Here endl is a stream manipulator, causing a linefeed to be inserted into stream. Thus values of a and b are displayed on 2 different lines.

3.6 COMMENTS in C++

Comments are important part of any program even though they are not mandatory to be given. Normally they are used to make program readable. The compiler ignores comments, so they do not add to the file size or execution time of executable program. They can be specified in C++ using // or /* */

/* the text to be commented */ is a single line comment,
while // is used as a multiline comment.

e.g.
```
#include<iostream.h>
void main()
{
    int a=10;
    //Following declaration will display value of a on screen
    cout<<"a="<<a;
}
```

So in this program the text "Following declaration will display value of a on screen" won't be compiled, so it will not be displayed on the screen.
Comments are useful to make program readable, if programming code is large enough.

3.7 SCOPE and VISIBILITY

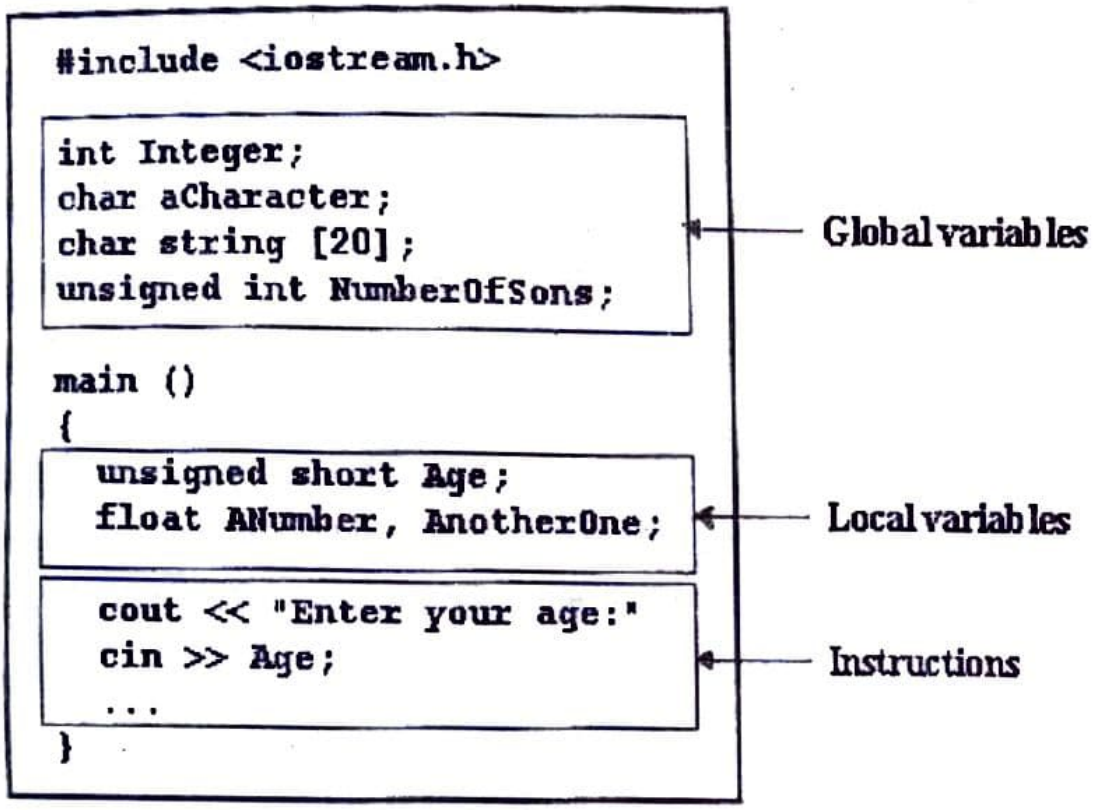
The scope or visibility of a variable determines whether it can be accessed from other functions in the same source file or in other source files. The position of an identifier within a block is also a factor in determining scope. In others words it is that part of a program over which the identifier can be seen by other identifiers and used by other identifiers.

External variables, defined outside any block and memory for these is allocated once. Even though the external variable is declared more than once, it only has memory allocated to it once. Variables can be declared as global variables, available to everything, these are also known as external variables. External variables are defined outside of any functions and retain their values during the course of a program.

Static variables are variables that are local to a function but which (unlike auto variables) retain their values between invocations of the function - these are internal static variables. The second kind of static are external or global variables that are local to a file but not a part of any function. The main use of these is to hide the variable so that other source code files don't see the variable. Static storage comes from a different memory allocation - it is allocated from a static area of memory so static variables within

functions can retain their values between calls to the functions. Static variables can be initialized to some value other than 0 but will only be initialized at the first call.

Automatic variable inside a function doesn't exist until the function is called. At that time the memory for the automatic variable is allocated dynamically from a memory stack and released when the function is finished. Auto variables are not initialized at start-up but each time they are called and do not retain their values.



Global variables can be referred anywhere in the code, within any function, whenever it is after its declaration.

The scope of the **local variables** is limited to the code level in which they are declared. If they are declared at the beginning of a function (like in `main`) its scope is the whole `main` function. This means that if in the example above, moreover than the function `main()` another function existed, the local variables declared in `main` could not be used in the other function and vice versa.

In C++, the scope of a local variable is given by the block in which it is declared (a block is a group of instructions grouped together within curly brackets `{}` signs). If it is declared within a function it will be a variable with function scope, if it is declared in a loop its scope will be only the loop, etc...

In addition to **local** and **global** scopes exists the external scope, that causes a variable to be visible not only in the same source file but in all other files which will be linked together with.

QUESTIONS

1.Select the correct alternatives.

a) C++ is a _____language
(procedural, object oriented, structured)
b)_____is a octal number
(076,23,0xAC)
c)_____is unary operator.
(+,-,++)
d)_____ is a keyword in C++.
(for ,int ,both)
e) The size of int is_____bits.
(2,6,16)
f) char is ____data type
(derived,built in,user defined)
g)_____variables are automatic variables by default.
(static ,local ,global)
h)_____is ternary operator.
(*, ?: , →)
i)_____ operator is used to give remainder after integer division.
(% , / ,both)
j)_____ is a bit wise operator
(&& , & , *&)
k)._____ operator resolves ambiguity between local and global variables.
(- , ::, ?:)
l)_____is a stream manipulator
(for, while, setw)

Answer:

a) object oriented b) 076 c)++ d) both e)16 f) built in g) local h) ? :
i) % j) & k):: l) setw

2.Answer the following:

1. Explain the structure of simple C++ program.
2. Explain significance of comments in C++.
3. What are different types of data types in C++?
4. What are variables? Explain the difference between declaration and initialization of variable.
5. What is constant? Explain different types of constants.
6. Explain different rules for a valid identifier.
7. What are manipulators? Explain different stream manipulators in C++.

8. Explain different bitwise operators.
9. Explain conditional operator.
10. What is significance of unary, binary, ternary operators? Explain with the example.
11. What is scope resolution operator? How is it used to solve ambiguity between local and global variables?
12. Explain automatic conversion.
13. What are different operators dealing with streams?
14. Explain the difference between prefix and postfix operator.
15. Expalin the significance of comments in C++? How they are specified?

Programs For Practice

1. Write a program in C++ to print product of 2 variables with postfix incremental values and prefix incremental value.

[**Hint:** prefix incremental product is; product = (++ number 1) * (++ number 2) Postfix incremental product is; product = (number ++) * (number 2 ++).]

2. Given the following declarations:

int a = 0, b = 1, c = -1;
float i = 2.5, j = 0.0

Evaluate the following and justify your answers:

i) a > b, ii) b / c < i, iii) a <= b >= c, iv) a <= I = = b, v) ++ a = = b ! = j * 2

Answers: i) 0 , ii) 1, iii) 1, iv) 1, v) 1

3. Given the following declaration

int a = 0, b = 1, c = -1
float p = 2.5, q = 0.0

Evaluate the following and justify your answer.

a) a && b, b) a < b && c < b, c) b + c !! !a, d) (a !! b) + (p !! ++ q)

Answers: a) 0, b) 1 (true), c) 1 (true), d) 2

4. Write the output of the following program

```
#include <iostream.h>
int a=20;
void  main()
{
  int a=10;
 cout<<"value of a is:"<<a;
 cout<<"now value of a is:"<<::a;
}
```

❑❑❑

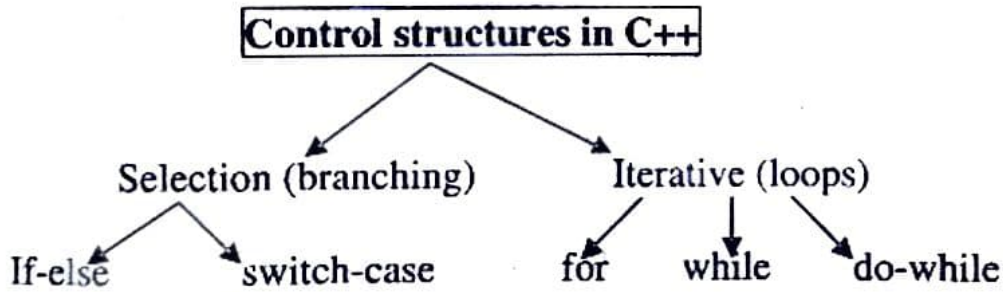
Up till now we have seen built in data types and related operators now, in this part, we are going to study few derived (compound) data types.

The flow of control:

When a programmer is building a program, it is good practice to break the program down into pieces that can be thought of independently. Once the program has been completed, we can think of its execution as being a series of these pieces that work together in a certain sequence. These pieces then pass the control of the program between each other. While one piece has the control, the other pieces are inactive. This is known as the flow of control in a program. For implementing it , C++ provides few control statements.

3.8 CONTROL STATEMENTS

Control Statements, are ways for a programmer to control what pieces of the program are to be executed at certain times. The syntax of Control statements are very similar to regular English, and are very similar to choices that we make every day. There are two basic types of control statements: **branching statements and loops. In branching statements** depending on a certain condition a certain series of events will be executed, while loops let the user to have certain part in the program to be repeated depending upon the terminating condition specified in the loop.



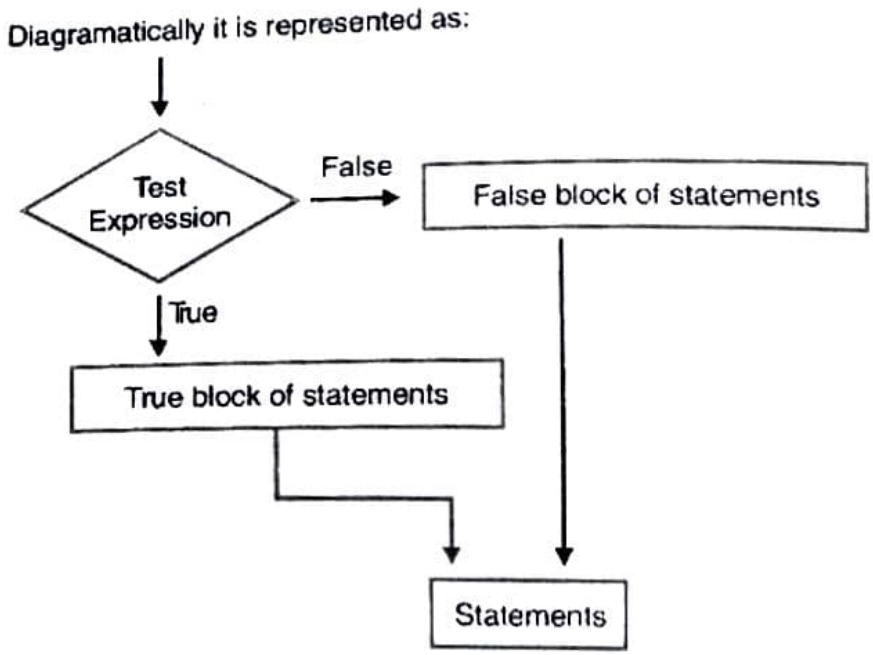
Branching Statements:

1. if statement :

The first type of branching statement we will look at is the if statement. An if statement has the form:

Syntax:

```
if(condition)
{
        //code to be executed if condition is true
}
else
{
        //code to be executed if condition is false
}
statements ....;
```



In an if statement, *condition* is a value or an expression that is used to determine which code block is executed, and the curly braces act as "begin" and "end" markers.

1. Program to find maximum between two numbers.

```
#include <iostream.h>
void main ( )
{
        int x=3,y=4;
        if(x>y)
                cout <<"x is greater than y"<<endl;
        else
                cout <<"x is smaller than y"<<endl;
}
```

Output is : x is smaller than y.

Now let's study how **multiple conditions** are combined using only if statement...

Compound conditions in if (multiple if's):
You can have any number of if statements in your program.
Following program combines multiple conditions in the if statement and o/p will be minimum of 3 numbers.

2. Program to find minimum of three numbers (using multiple if).

```
#include <iostream.h>
void main ( )
{
        int a, b, c;
        cout <<"Enter 3 numbers:";
        cin >> a >> b >> c;
        if (a <b && a < c)
        cout <<"Minimum is:" << a << endl;
        if ( b < a && b < c)
        cout <<"Minimum is:" << b << endl;
        if(c < a && c <b)
        cout <<"Minimum is:" << c << endl;
}
```

Output is: Enter 3 numbers: 3 2 6
Minimum is: 2

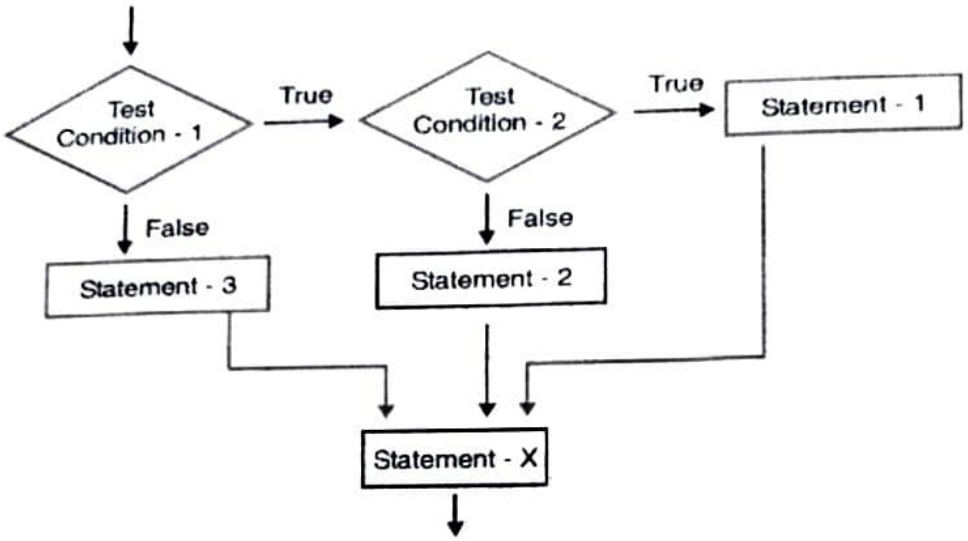
Many times, there arise a situation, where you need to work on **interdependent conditions**, C++ handles this by means of **nesting**(embedding 2 or more control structures/constructs)

Nested selection statements (nested if else)

Syntax :-

```
if (test-condition 1)
{
        if (test condition 2)
        {
                statement – 1;
        }
        else
        {
                statement – 2;
        }
}
else
{
        statement – 3;
}
statement – x;
```

Diagrammatically it can be represented as:



You can embed any number of if statements within outer if statement. This is called nested ifs. The above program for finding minimum of three numbers can also be done using nested if-else statements.

3. Program to find minimum of three numbers (using nested if).

```
#include <iostream.h>
void main ( )
{
        int a, b, c;
        cout <<"Enter 3 nos:";
        cin >> a >> b >> c;
        if (a < b)
        {
                if(a<c)
                        cout <<"Minimum is:" << a << endl;
                else
                {
                        if ( b < c)
                                cout <<"Minimum is:" << b << endl;
                }
        }
        else
                cout <<"Minimum is:" << c<< endl;
}
```

Output is
Enter 3 numbers: 3 2 6
Minimum is: 2

Replacing if...else by conditional operator
In the last chapter we have already seen ternary conditional operator. It can be used to replace if-else construct.

4. Program to find maximum of two numbers (using conditional operator).

```
#include <iostream.h>
void main ( )
{
        int no1, no2, max;
        cout <<Enter 2 nos:";
        cin >> no1 >> no2;
        if (no1 > no2)
                cout <<"Maximum is:" << no1 << endl;
        else
                cout <<"Maximum is:" << no2 << endl;
        max = ((no1 > no2) ? no1 : no2);
        cout <<"Maximum is:" << max << end;
}
```

Program Explanation: Here conditional operator? : is used to determine maximum of given numbers. No.1 will be assigned to max if no.1 is greater than no.2 which is equivalent to if part while, if no.1 is less than no.2, no.2 is assigned to max which is equivalent to else part.

Note: **A common programming error:**

```
#include <iostream.h>
void main ( )
{
        int no;
        cout <<"Enter the number:";
        cin >> no;
        if (no = 50)
                cout <<Number is:" << no;
        else
                cout <<"Number is not equal to" << no;
}
```

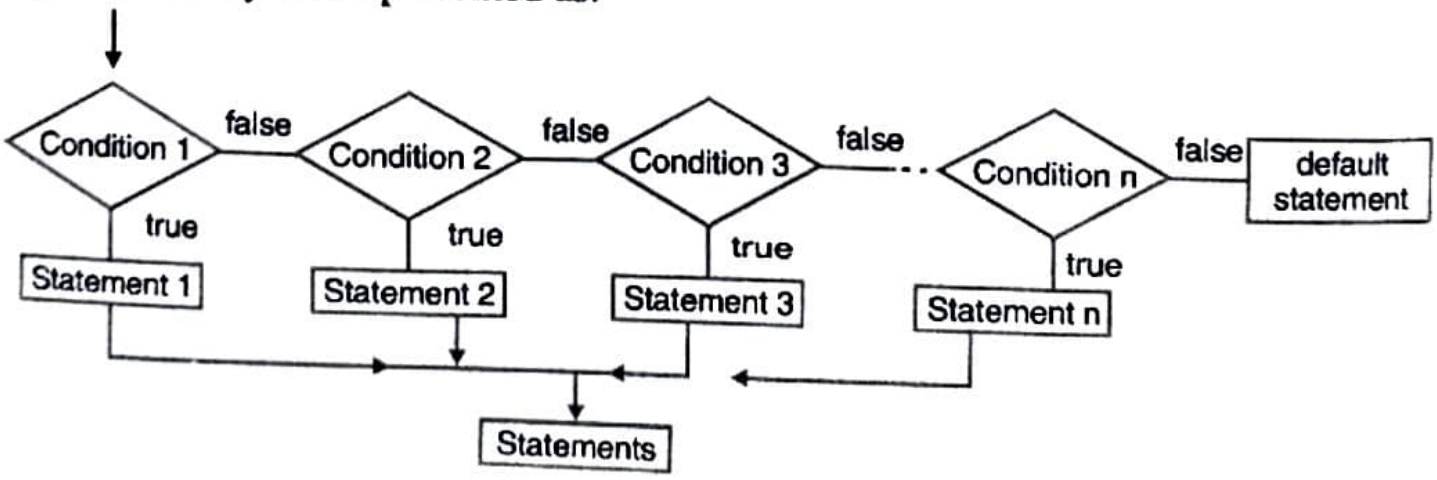
Output is : Enter the number: 70
Number is 50

This program will result into logical error. Here expression (no = 50) assigns value 50 to no which is treated as true condition and there if part gets executed.
It is corrected as if (no = = 50)
Now let's study how multiple conditions can be handled++ will provide you to use either multiple else-if s (called **else-if ladder**) or **switch-case** construct.

Else-if ladder:
This is normally to be used in making multipath decisions.
Syntax:

```
if (condition 1)
        statement 1;
elseif (condition 2)
        statement 2;
elseif (condition 3)
        statement 3;
        ....
elseif (condition n)
        statement n;
else
        default statement;
statements.....;
```

Diagrammatically it is represented as:



Here all conditions are tested and the respective block is evaluated, corresponding to satisfaction of a condition.
If none of the conditions returns true, then else part (which is equivalent to default case in switch ... case construct) is evaluated.

Following program uses else-if ladder control structure to decide the grade of the student.

5. Program to find class according to % marks (using multiple path)

```
#include <iostream.h>
void main ( )
{
        int total;
                cout <<"Enter total marks:";
        cin>>total;
        if (total > 100 && total <= 0)
                cout << "Invalid";
        elseif (total >= 70 && total < 100)
                cout << "Distinction";
        elseif (total >= 60 && total < 70)
                cout << "First class";
        elseif (total >= 50 && total < 60)
                cout << "Second class";
        else
                cout << "Fail !";
}
```

Output is : Enter total marks:85
Distinction.

Program Explanation: In this code, by using else-if ladder, total is checked against various conditions and accordingly grade of the student is declared. So here o/p will be **distinction** if total marks obtained by student is in between 70 and 100, **first class** when it is in the range of 60 to 70, **second-class** if it is in the range of 50 to 60. To impose both the conditions, we have made use of && operator.

Two extreme conditions are also checked i.e. if total is –ve no. or if it exceeds 100, 50 program will print **invalid** and simply terminates.

If none of the above conditions, net, then simply **fail** will be displayed on the screen.

2.switch statement:

The next branching statement is called a switch statement. A switch statement is used in place of many if statements. The **switch statement** allows a programmer to compound a group of **if statements (else-if ladder)**, provided that the condition being tested is an integral constant i.e. either integer or character. It is also called as **multiple branch selection statement.**

Syntax: switch(*integer_val*)

```
{
        case value_1 : // code to execute if integer_val is value_1
```

```
                    break;
                    ...
        case value_n // code to execute if integer_val is value_n
                    break;
        default:    // code to execute if integer_val is none of the above
    }
```

Here expression returning an integral constant is matched against all cases and only matching case is evaluated. No two cases are equal. The **default** clause is optional, but it is good programming practice to use it. The default clause is executed if none of the other clauses have been executed. This proves to be more efficient than using multiple if statements when we need to select between many conditions/alternatives. But the difference between if and switch construct is ,switch can test only for equality ,unlike if which tests for any relational or logical expression.

Program5: Deciding the day of a week using switch-case construct.

```
#include<iostream.h>
void main()
{
    int day;
    cout<<"Enter number in the range of 1 to 7 to get corresponding day:";
    cin>>day;
    switch(day)
    {
        case 1: cout<<"Today is Sunday";
                break;
        case 2: cout<<"Today is Monday";
                break;
        case 3: cout<<"Today is Tuesday";
                break;
        case 4: cout<<"Today is Wednesday";
                break;
        case 5: cout<<"Today is Thursday";
                break;
        case 6: cout<<"Today is Friday";
                break;
        case 7: cout<<"Today is Saturday";
                break;
        default:cout<<"Not a valid day";
                break;
    }
}
```

Output is: Enter the number in the range of 1 to 7 to get the corresponding day :3
Today is Tuesday

If none of the case is matched i.e. if you give any number other than 1 to 7 default case is evaluated and output is : Not a valid day

Significance of break keyword: --**break** keyword is optional to be given, But if you do not specify break keyword, compiler won't flag any error, but all the case statements will be executed following the matching case. The **break** keyword means, "jump out of the switch statement, and do not execute any more code."

3.Unconditional control structure (goto)

Up till now we have seen flow of control going conditionally. But if you want to go at any point in the program, unconditionally C++ lets you to do this by using **goto** construct.**goto** makes use of label to locate the place at which program control should be passed. A label is not a keyword but any valid variable name. Also it can be given anywhere and to the any statement in the program and may be placed above or below goto construct. It can also be used to construct a loop. But it is not a good programming practice to use goto in your program.

Syntax : 1)
```
goto label ;
...
...
label : statements.....;
```
2)
```
label : statements.....;
...
...
goto label ;
```

e.g.
```
#include<iostream.h>
void main()
{
        int x=100;
        abc : x--;
              if(x>0)
                goto abc;
}
```

3.9 Iterative(repetitive)control structures (Loops):---

1.for statement :--

syntax: **for**(initial_value ; test condition ; step)
```
{
        // Code to execute inside loop
}
```

Here **initial_value** sets up the initial value of the loop counter.
test condition is the condition that is tested to see if the loop is executed again.
step describes how the counter is modified(incremented or decremented) on each execution of the loop. And code inside the for loop is executed till condition is getting satisfied

6. Program to find addition of first 10 numbers using for loop.

```
#include<iostream.h>
void main()
{
        int total=0;
        for (int i=1; i < 11; i++)
        {
        total = total + i;
        }
        cout<<"sum of first 10 numbers is--:"<<total;
}
```

Output is: sum of first 10 numbers is--: 55.
Program explanation : Here i is acting as a loop control variable or counter. For each iteration value of i is getting modified, in this case it is incremented by 1.So finally loop is aborted when value of i will reach value 11.You can use same variable many times in different loops provided scope of that variable is finished in the first loop, then only you can use it in second.

Nesting Of For Loop

Nesting means, loop inside a loop. You can nest for loop inside the other for loop.

7. Program to prints multiplication table for numbers 1 to 5 (using for Loop).

```
#include<iostream.h>
#include<iomanip.h>
void main()
{
    for(int count1=1;count1<=5;count1++)
    {
        for(int count2=1;count2<=5;count2++)
            cout<<setw(5)<<count1*count2;
        cout<<endl;
    }
}
```

Output is :

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

Program explanation: In this program, second for loop is nested inside the first for loop. Each iteration of first for loop (Count 1) will print one row of the multiplication table and second for loop (count 2) iterates, from 1 to 5 for each of the iteration of first for loop.
In this program we have used a stream manipulate setw, to set the width of the output field for each number printed.
Here expression setw (5) means to set the output field width to 5 columns. It is a stream manipulator. For this we have included <iomanip.h> file.
Here endl is given outside the second for loop and inside the first for loop so that multiplication table for each of the number is printed on a separate line.

2.while statement

Syntax : while(*condition*)

```
{
        // code to execute
}
```

Condition is a Boolean statement that is checked each time after the final "}" of the while statement .If the *condition* is true then the while statement executes again. If the *condition* is false, the while statement does not execute again.

7. Program to print sum of first 10 numbers using while loop.

```
#include<iostream.h>
void main()
{
        int total=0,i=1;
        while(i < 11)
        {
                total = total + i;
                i++;
        }
        cout<<"sum of first 10 numbers is--:"<<total;
}
```

Output is: sum of first 10 numbers is--: 55.

Now let's study how while loop is used to repeat a computation, till, the condition, imposed by user.

In a single run of the program, you can do any number of computations till condition given in while is true.

Following program prints the square root of given numbers using standard **(built in) library** function **sqrt ( ) present in <math.h >file** , till number entered is greater than 0.

8. Program to print the square root of given numbers using standard (built in) library function

```
#include<iostream.h>
#include<math.h>
void main()
{
        double number;
        cout<<"Enter a positive number:";
        cin>>number;
        while(number>0)
        {
                cout<<"square root of"<<number<<"is"<<sqrt(number)<<endl;
                cout<<"Enter the next number or enter 0 to quit:";
                cin>>number;
        }
}
```

Output

Enter a number: 81
Square root of 81 is: 9
Enter next number or enter 0 to quit: 4
Square root of 4 is: 2
Enter next number or enter 0 to quit: 0

Program Explanation:

In this program square root of a given number is calculated repeatedly using while loop till user presses 0 or some negative number. When 0 is given, loop is not executed since condition won't satisfy and program is terminated

3.Do.....while statement.

When you want to execute loop at least once, then instead of while, you can choose do...while loop. In this loop, also test condition is checked at last, so loop is executed at least once even though condition is not getting satisfied while entering the loop itself.....

Syntax: **do**

```
{
    // code to execute
} while(condition) ;
```

e.g .Same program we have done using for and while is now done using do...while loop.

9. Program to print sum of first 10 numbers using (do.... while) loop.

```
#include<iostream.h>
void main ()
{
    int total=0,i=0;
    do
    {
        total = total + i;
        i++;

    }
    while(i < 11);
    cout<<"sum of first 10 numbers is--:"<<total;
}
```

Output is : sum of first 10 numbers is--:55.

3.10 Use of break and continue:

Break and continue both are the keywords. break statement causes rest of the statements in the loop's block, jumping immediately to the next statement outside the loop, while continue statement transfers execution to the next iteration of the loop. It continues the loop after skipping the remaining statements in its current execution.

Aborting an Infinite Loop

The loop will continue to execute forever, if there is no way provided for termination, called infinite loop. Break or exit(0) function ,are used to terminate this infinite loop. So if you don't provide a mechanism for termination (like exit (0) or break) then you will get infinite loop. So to stop in between press (ctrl + c) keys on keyboard.

10. Program to use break statement to terminate infinite loop.

```
#include<iostream.h>
void main()
{
        int n,i=1;
        cout<<"Enter a positive integer:";
        cin>>n;
        long sum=0;
        while(true)
        {
                if(i>n)
                        break;
                sum+=i;
                i++;
        }
        cout<<"sum of first "<<n<<"integers is:"<<sum;
}
```

Output is : Enter a positive integer:10
sum of first 10 integers is:55
Enter a positive integer: -10
sum of first 10 integers is:0

Program Explanation: Note that, the control condition in the while loop itself is true, which means it continues forever. The loop is aborted when value of i (getting modified in loop) exceeds the given number. In this program suppose negative value is given, value of n becomes negative and since i is initialized to 1 , so if condition over here is getting false and break will cause termination of loop and immediately display sum which is initialize to 0,therefore you will get message sum of first 10 integers is:0 to be displayed on the screen.

One advantage of using a break statement inside a loop is that it causes the loop to terminate immediately without having to finish executing the terminating statements in the loop block.

Thus break can be used to terminate the loop, "breaking out" of the iteration at that point and it provides extra flexibility n the control of loops. Normally a while loop, do...while or a for loop will terminate only at the beginning or at the end of the complete sequence of statement in the loop is block. But break statement can be placed anywhere among the other statements within the loop; so that it can be used to terminate a loop anywhere from within the loop's block.

11. Program to use break and continue

```
#include<iostream.h>
void main()
{
        int number;
        for(; ;)                    //infinite loop.
        {
                cout<<"Enter  the number:";
                cin>>number;
                if(number % 4= = 0 )
                        continue;
                if(number % 5 = = 0 )
                        break;
                cout<<"At the end!!!";
        }
        cout<<" Completely outside !!! ";
}
```

Output is:

Enter the number:8
Enter the number:10
Completely outside!!!
Enter the number:7
At the end!!!

Program Explanation: -- When number entered is 7, both of the if conditions are false and displays At the end !!!
While, when number entered is 8, for next iteration, the first if condition is true (8 is completely divisible by 4), so control skips over the rest of the statements inside the loop and jumps immediately to the top of the loop again to continue with its next iteration.
Now, when number entered is 10, second if condition gets true so control breaks out of the loop and jumps immediately to the first statement that follows the loop displaying Completely outside !!!

Use Of **Exit (0)** Function

The exit ( ) function provides another way to terminate a loop.
When it is executed, it terminates the program itself.

12. Program to Generate Fibonacci Series

```
#include<iostream.h>
void main()
{
        cout<<"Enter a positive integer:";
        cin>>limit;
        cout<<"Fibonacci numbers less than<< limit<<"are"<<":\n0,1";
        long term1=0,term2=1;
        for(; ;)
        {
                long term3=term1+term2;
                if(term3>limit)
                        exit(0);
                cout<<","<<term3;
                        term1=term2;
                        term2=term3;
        }
}
```

Output is:

Enter a +ve integer: 300
Fibonacci numbers less than 300 are:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233
Here in this program, since there are no statements following the loop, terminating the loop is same as terminating the program.

3.11 FUNCTIONS IN C++

To make larger programs manageable, programmers modularize them into subprograms or small modules called functions. Functions are intended to do a particular task. They can be compiled and tested separately and can be reused in other programs. Thus complexity of the entire program can be divided into simple subtasks, these subtasks are easier to write, code and debug. Standard C++ library is a collection of pre-defined functions called **library or in-built functions** and other program elements, which are accessed through header files. While user can also write his/her code to have a function called **user-defined function**. Thus Functions let us create logical groupings of code.
The typical function layout is :

```
return-type function_name(list of parameters)
{
        local variable declarations;
```

```
        body of the function;
        return statement;
}
```

Where the value returned by the return statement must match the return type. (If function never returns a value you may leave out return statement, and use a return type of **void**.) A C++ function should have a **prototype**, declaring its arguments and return type.

13. Program to display area of circle.

```
double area(double);                    //function prototype.
#include<iostream.h>
void main()
{
        double r,a;
        cout<<"Enter radius:";
        cin>>r;
        a=area(r);              //function call.
        cout<<"Area of circle is:"<<a;
}
double area(double r)           //function definition
        {
        double a;
        a=3.14*r*r;
        return a;
}
```

Output is: Enter radius: 3
Area of circle is: 28.26

Same program can be done by defining return type of the function to be void.

13. Program to display area of circle (using return type function)

```
void area(double);              //function prototype.
#include<iostream.h>
void main()
{
        double r,a;
        cout<<"Enter radius:";
        cin>>r;
        area(r);                //function call.
}
```

```
void area(double r)              //function definition
{
        double a;
        a=3.14*r*r;
        cout<<"Area of circle is:"<<a;
}
```

3.12 Default arguments

Functions taking values are called arguments or parameters. They are not mandatory to be given .When programmer initializes values while specifying arguments, it is said that function takes default arguments. They even can be specified in the prototype.

14.Program to demonstrate use of default arguments.

```
void add( int a, int b = 0,int c=6 );
#include<iostream.h>
void main()
{
        add(3);
        add(3,5);
        add(1,2,3);
}
void add(int a,int b,int c)
{
        cout<<(a+b+c);
}
```

Output is : 9 14 6

Program Explanation: --When first add function is invoked with parameter 3,output is 9(since it expanded as 3+0+6)
When second add function is invoked with parameter 3 and 5,output is 14(since it expanded as 3+5+6)
When third add function is invoked with parameter 1,2,3 respectively, output is 6(since it expanded as 1+2+3) Here values of b and c specified in prototype as default arguments are overridden by those which are specified during function call.

Rule: The default parameters must be the in the last position.

e.g. function(int a = 0, int b = 10, int c); // compilation error as **missing default parameter for parameter 3**
But , function (int a, int b = 0, int c = 10); // OK .No error !!

3.13 TECHNIQUES USED TO PASS VARIABLES INTO C++ FUNCTIONS

1. Pass by value
2. Pass by reference with a pointer
3. Pass by reference using a reference

1. Pass by value:

The function receives a copy of the variable. This local copy has scope, that is, exists only within the function. Any changes to the variable made in the function are not passed back to the calling routine. The advantages of passing by values are simplicity and that is guaranteed that the variable in the calling routine will be unchanged after return from the function. There are two main disadvantages. First, it is inefficient to make a copy of a variable, particularly if it is large such as an array, structure or class. Second, since the variable in the calling routine will not be modified even if that's what is desired, only way to pass information back to the calling routine is via the return value of the function. Only one value may be passed back this way.

15. Program to demonstrate use of pass by value.

```
void passbyvalue(int);
#include<iostream.h>
void main()
{
        int i=4;
        passbyvalue(i);
        cout << "i is :" << i << " \n";
}
void passbyvalue(int i)
{
i = i + 1;
}
```

Output is : i is 4.

Program Explanation: Since i was passed by value, the local copy in the function gets incremented. The copy in main remained unmodified and still i will be 4.

2. Passing by reference using a pointer:

A pointer to the variable is passed to the function. The pointer can then be manipulated to change the value of the variable in the calling routine. The function cannot change the pointer itself since it gets a local copy of the pointer. However, the function can change the contents of memory, the variable, to which the pointer refers. The advantages of passing by reference are that any changes to variables will be passed back to the calling routine and that multiple variables can be changed.
(For this refer **pointers** later in this chapter).

.16. Program to demonstrare use of passing by refernce using pointer.

```
void passbyptr(int *newi);
#include<iostream.h>
void main()
{
        int i=4,*ptr;
        ptr = &i;       //set the pointer to the address of i
        passbyptr(ptr);
        cout<< "i is :"<<i<<"\n";
}
void passbyptr(int *i)
{
        *i = *i + 1;
}
```

Output is : i is 5

Since i was passed by reference, the function received a pointer to the variable. By properly dereferencing, i was incremented to 5.

3. Passing by reference using a reference:

A reference in C++ is an alias to a variable. Any changes made to the reference will also be made to the original variable. When variables are passed into a function by reference, the modifications made by the function will be seen in the calling routine. Since no local copies of the variables are made in the function, this technique is efficient. Additionally, passing multiple references into the function can modify multiple variables.
(see the topic **reference, later** in this chapter)

17. Program to demonstrare use of passing by reference using reference

```
void passbyref(int &);
#include<iostream.h>
void main()
{
        int i=4;
        int &m=i;              //reference to i.
        passbyref(m);
        cout << "i is :"<<i<<"\n";
}
void passbyref(int &i)
{
        i = i + 1;
}
```

Output is: i is 5.

Since i was passed by reference, the variable in the calling routine is modified.
By default, parameter is called pass-by-value, while Pass-by-reference, or reference parameters, allows the function to alter the value of a variable passed to it. Pass-by-reference is achieved by placing an ampersand before the parameter identifier in the function header. Arrays and structures are automatically passed by reference. (refer **arrays** later in this chapter)

main function in C++:--
Every program has main() function. Valid definitions of main() have return type 'int' (explicit or implied).This is the case in both Standard C and in C++. If you write, for example, void main(){ } ,the compiler is obligated to diagnose an error in C and C++.The return value from main() is supposed to be supplied to the calling environment .Some systems appear not to care whether you return a value, but some report error conditions based on the garbage value which main() otherwise implicitly returns.

3.14 FUNCTION OVERLOADING

Overloading is one of the features of **object-oriented languages.**
C++ provides 2 types of overloading:
a) Function overloading b) operator overloading
Now let's study function overloading. A function is said to be overloaded when it is declared more than once in a program. Overloading allows a set of functions that perform a similar operation to be collected under the **same name**. It provides **alternate interfaces**

(different implementations) to the same function. When there are several declarations of the same function, the compiler determines which function should be called by examining the argument signature (number of arguments, or data type of the arguments or both) of the function call.

So it is nothing but one of the ways of representing polymorphism (compile time)in object oriented technology.(In next year we are going to study polymorphism as one of the object oriented features)

18.Following program calculates the area of circle and rectangle using overloaded function area

```
double area (int);
int area (int,int);
#include<iostream.h>
void main()
 {
        int r,l,b;
        double areacircle, arearect;
        cout<<"Enter Length and breadth:";
        cin>>l>>b;
        arearect=area(l,b);
        cout<<"area of rectangle is:"<<arearect<<endl;
        cout<<"Enter radius:";
        cin>>r;
        areacircle=area(r);
        cout<<"area of circle is:"<<areacircle;
 }
double area(int r)
 {
        double area1;
        const double pi=3.14;
        area1=(pi*r*r);
        return area1;
 }
int area(int l,int b)
 {
        int area2=l*b;
        return area2;
 }
```

Output is: Enter Length and breadth:3 5
area of rectangle is:15
Enter radius:3
area of circle is:28.26

Here area function is said to be overloaded. Since it takes different arguments (in number). The requirement is arguments of functions should either differ in number or type or both. Return type of the function doesn't matter.

3.15 INLINE FUNCTIONS

An **inline** function is a function whose code gets inserted into the caller's code stream. Thus, when a function is declared as inline, it tells the compiler to replace each call to the function with explicit code for the function.

Like a #define macro, inline functions improve performance by avoiding the overhead of the call thus inline functions are new design approach to speed up program execution.

They save memory space, as the calls to the same code need not be duplicated in memory.When you declare an inline function, it looks just like a normal function:

void function1(int x, char y);

But when you define an inline function, you prepend the function's definition with the keyword **inline**, and you put the definition into a header file:

```
inline void function1(int x, char y)
{
        //function code.....
.}
```

But note that overuse of inline functions should be avoided since it can cause size of the code getting larger and that might cause the operating system to *thrash*, i.e. use of inline function can cause negative side effects also, like it can limit the portability of your code across platforms.

19. Program to demonstrare use of inline function

```
int square (int);
#include<iostream.h>
void main()
{
        int number=5;
        cout<<"square of the given number is:"<<square(number);
}
inline int square(int a)
{
        return (a*a);
}
```

Whenever inline function is called, the code of the function is inserted by the compiler, so, above program is equivalent to :--

```
void main()
{
        cout<<"square of the given number is:"<< (5*5);
}
```

3.16 RECURSION (Recursive functions)

When function, calls itself within its body, it is called as **recursive** function. But note that the value of the argument of the passed argument must be different. Recursion is also called as **circular definition.** Note that all the functions can't be made recursive.

20.Following program calculates factorial using recursion.

```
int factorial(int n);
#include<iostream.h>
void main()
{
        int n,f;
        cout<<"Enter the no,of which factorial is to be found:";
        cin>>n;
        f=factorial(n);
        cout<<"Factorial is:"<<f;
}
int factorial(int n)
{
        int fact=1;
        if(n==1)
                return 1;
        else
        {
                fact=factorial(n-1)*n;
                return (fact);
        }
}
```

Output is : Enter the no, of which factorial is to be found : 3
Factorial is: 6

Program Explanation: -- In this case **factorial** is the function, which is made recursive. We know n ! = n*(n-1)*(n-2)*……..3*2*1

Alternatively it can be represented as n*(n-1)!
Which can also be written as n*(n-1)*(n-2)! ...etc.
That means you are repeating calculation of factorial with different parameters (with one argument less) every time.
Thus in this case recursion is used.

Tracing of the above program is as follows...

Factorial(3) execution trace.

Call	n	n = = 0	n * Factorial(n-1)	returns
First	3	0	3 * Factorial(2)	6
second	2	0	2 * Factorial(1)	2
Third	1	0	1 * Factorial(0)	1
fourth	0	1		1

3.17 ARRAYS IN C++

An **array** consists of a set of objects (called its elements), all of which are of the same type and are arranged **contiguously** in memory. Each element is identified by an **index,** which denotes the position of the element in the array. The number of elements in an array is called its **dimension.** The dimension of an array is fixed and predetermined; it cannot be changed during program execution. Arrays are declared with the number of storage locations indicated, but array element references start at zero.
Arrays are suitable for representing composite data, which consist of many similar, individual items. Examples include: a list of names, tables of students roll number, or the monthly transactions for a bank account.
Note that data type of all the elements in the array is same.
An array variable is defined by specifying its dimension and the type of its elements.
e.g. int rollnum[10];

Indexing the array accesses the individual elements of the array. The first array element always has the index 0. Attempting to access a nonexistent array element leads to a serious **runtime error** (called '**index out of bounds error**).

Caution:--Using an index value outside the array bounds can have unpredictable results. You are actually requesting direct access to memory that is not within the confines of the array! Since C++ by default has no **bounds checking,** the programmer must take on this responsibility. If a program that uses arrays exhibits strange behavior, the programmer should check carefully to make sure all array element references are within bounds.

Processing of an array usually involves a loop that goes through the array element by element.
Like other variables, an array may have an initializer. Braces are used to specify a list of comma-separated initial values for array elements.
e.g. int numbers[3] = {1, 2,3};
Initializes the three elements of numbers to 1,2,3 respectively.

Note: Array can be initialized but can't be assigned.

```
e.g.    int a[3]={1,2,3};
        int b[3]={4,5,6};
        b=a;                    //ERROR!!:arrays can't be assigned.
```

21.Following program accepts and displays array elements

```
#include<iostream.h>
void main()
{
        int a[5],i;
        cout << "This Program is to input 5 numbers and display them";
        cout << "Enter 5 numbers:";
        for (i=0; i<=5; ++i)
        {
                cin >> a[i];
        }
        for (i=0; i<=5; ++i)
        {
                cout <<"Element number"<< i+1 <<":"<< "\t"<< a[i];
        }
}
```

Output is : This Program is to input 5 numbers and display them
Enter 5 numbers : 1 2 3 4 5
Element number 1 : 1
Element number 2 : 2
Element number 3 : 3
Element number 4 : 4
Element number 5 : 5

Array Of Characters (strings):--

A **C++ string** is simply an array of characters.(string literal)

e.g. char str[] = "HELLO";

defines str to be an array of six characters: five letters and a null character. The compiler inserts terminating null character.

Alternatively it can also be defined as

char str[] = {'H', 'E', 'L', 'L', 'O'};

A string literal is a **sequence of characters** enclosed within **double quotes**. The storage class of a string literal is **static**. A compiler initializes the memory occupied by a string literal with the specified character sequence. A null terminator (which denotes the end of the string) is also included. The size of the string literal is the number of characters enclosed within the double quotes plus one for the null terminator.

The compiler concatenates adjacent string literals.

e.g. int i = sizeof("i++"); // i = = 4

char arr[] = "i++"; // arr is a char array of size 4

It is easy to calculate the dimension of an array using the sizeof operator.

For example, given an array arr

so the dimension of arr is: **sizeof**(arr) / **sizeof**(datatype)

While there was no built-in string type in the original definition of C++, there were a number of strings processing functions provided as part of the standard library and declared in the header file 'string.h'. The more common of these include:

String concatenation :--

char* strcat(char *dest, const char* src);

appends src to dest and returns dest.

String comparision :--

int strcmp(const char* s1, const char* s2);

compares the two strings, s1 and s2.

If s1 < s2 (if s1 is before s2 alphabetically), a value less than 0 is returned.

If s1 = = s2, the value 0 is returned.

If s1 > s2), a value greater than 0 is returned.

String copy:-- The strcpy function copies its second argument to its first, character by character, including the final null character.

char* strcpy(char* dest, const char* src);

copies src to dest and returns dest

String length:-- The strlen function counts the characters in its string argument up to (but excluding) the final null character

int strlen(const char* s);

returns the length of s, not counting the null character.

Passing array to the function: -

Array can be passed to the function as a parameter,. When array is passed to the function, in the call to the function, we provide the **name** of the array, which is nothing but **address of the first element** in array, the function uses this address to access the array contents. The size of the array can also be specified,

22.Following program passes array to the function and calculate sum of its elements

```
void sum1(int[],int);
#include<iostream.h>
void main()
{
        int a[]={1,2,3,4};
        int size=sizeof(a)/sizeof(int);
        cout<<"sum of the array elements is:";
        sum1(a, size);
}
void sum1(int a[],int size)
{
        int sum=0;
        for (int i=0; i<=size; i++)
        {
                sum+=a[i];
        }
        cout<<sum;
}
```

Output is : sum of the array elements is: 10

Multidimensional Arrays

An **array of arrays** is referred to as multidimensional array, An array may have more than one dimension (i.e., two, three, or higher). The organization of the array in memory is still the same (a contiguous sequence of elements), but the programmer's perceived organization of the elements is different.

e.g. int twoDim[3] [2] = { {1,11}, {2,12}, {3,13}};

The above declared is 2-d array that is viewed as a one-dimensional array of one-dimensional array.

23.Following program illustrates reading and printing 2-d array:--

```
#include<iostream.h>
void main()
{
        int arr[3][3];
        int row,col;
        cout<<"Enter elements of matrix:";
        for(row=0;row<3;row++)
                for(col=0;col<3;col++)
                        cin>>arr[row][col];
        cout<<"Elements in the matrix are:"<<endl;
                for(row=0;row<3;row++)
                {       for(col=0;col<3;col++)
                                cout<<"\t"<<arr[row][col];
                                cout<<"\n";
                }
}
```

Output is : Enter elements of matrix: 1 2 3 4 5 6 7 8 9
Elements in the matrix are:
1 2 3
4 5 6
7 8 9

3.18 POINTERS IN C++

Pointer is simply the **address of a memory location** and provides an indirect way of accessing data in memory. A pointer variable is defined to **'point to'** data of a specific type.

```
e.g.    int     *ptr1;      // pointer to an int
        char    *ptr2;      // pointer to a char
```

The value of a pointer variable is the address to which it points.(i.e. address of the memory location where variable is stored)

```
e.g     int variable1;          we can write:
        ptr1 = &variable1;
```

The symbol **&** is the **address operator**; it takes a variable as argument and returns the memory address of that variable. The effect of the above assignment is that the address of

variable1 is assigned to ptr1. Therefore, we say that ptr1 points to value of variable1. Following figure illustrates this diagrammatically.

A simple integer pointer.

Given that ptr1 points to variable1, the expression
*ptr1 **dereferences** ptr1 to get to what it points to, and is therefore equivalent to variable1.The symbol * **is the dereference operator**; it takes a pointer as argument and returns the **contents of the location** to which it points.
In general, the type of a pointer must match the type of the data it is set to point to.
A pointer may be cast (type converted) to another type. For example,

```
ptr2 = (char*) ptr1;
```

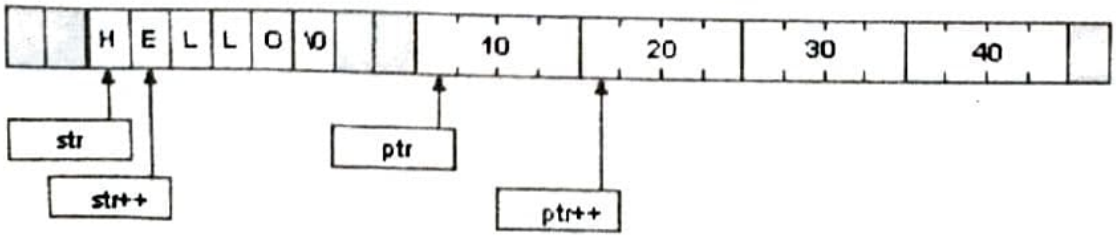
converts ptr1 to character pointer before assigning it to ptr2.
Regardless of its type, a pointer may be assigned the value 0 (called the null pointer).

Pointer Arithmetic

In C++ one can add an integer quantity to or subtract an integer quantity from a pointer. This is frequently used by programmers and is called pointer arithmetic. Pointer arithmetic is not the same as integer arithmetic, because the outcome depends on the size of the object pointed to. For example, suppose that an int is represented by 4 bytes. Now, given

```
char *str = "HELLO";
int  numbers[] = {10, 20, 30, 40};
int  *ptr = &numbers[0];                    // pointer to first element
```

str++ advances str by one char (i.e., one byte)(pointer will advance by the number of bytes of its data type), so that it points to the second character of "HELLO", whereas ptr++ advances ptr by one int (i.e., four bytes) so that it points to the second element of numbers. Following figure illustrates this diagrammatically.



It follows, therefore, that the elements of "HELLO" can be referred to as *str, *(str + 1), *(str + 2), etc. Similarly, the elements of numbers can be referred to as *ptr, *(ptr + 1), *(ptr+ 2), and *(ptr + 3).

Pointer comparison:--You can compare two pointers of compatible type for equality (or inequality).

3.19 Reference variables in C++:--allows two variable names to address the same memory location. The following example shows the technique in its simplest form. It should be noted that a reference does not create a copy of a variable, but merely a **symbolic alias** for it.

24.Following program demonstrates use of reference variable.

```
#include<iostream.h>
void main()
{
        int var;
        int &reference_var = var;
        var = 100;
        cout << " original variable is = " << var<< endl;
        cout << " reference variable is = " << reference_var << endl;
}
```

Output is: original variable is = 100
reference variable is = 100

A reference must always be initialized when it is defined: it should be an alias for something. It would be illegal to define a reference and initialize it later.

```
double &var1;          // illegal: reference without an initializer
var1 = num2;
```

you can also initialize a reference to a constant. In this case a copy of the constant is made (after any necessary type conversion) and the reference is set to refer to the copy.

```
int &n = 1;            // n refers to a copy of 1
```

The most common use of references is for function parameters. Reference parameters facilitates the pass-by-reference style of arguments, as opposed to the pass-by-value style

25.Following program demonstrates passing reference variable to function.

```
#include <iostream.h>
void fun1(int& ref);
void main()
{
        int var=3;
        fun1(var);
        cout << var;
}
void fun1(int& ref)
{
    ref = 2*ref;
}
```

This program prints "6" (2*ref doubles the value of variable referenced by ref,i.e var).

3.20 ARRAYS AND POINTERS

Arrays and pointers are very closely related C++ compiler interprets the name of the array as an address of the first element. You can traverse the whole array with the help of pointers. Once the pointer is initialized to the starting address of the array, it works like an index of the array.

26. Following program demonstrates traversing array with the help of pointers:-

```
#include<iostream.h>
void main()
{
        int a[5];
        int *p;                 //Pointer is initialized
        p = a;                  //it is equivalent to p=&a[0];
        cout << "Enter elements of array:";
        for (int i = 0; i<5; i++)
        {
                cin>>a[i];  //it is equivalent to cin>>*p;
        }
        cout <<"Array elements displayed using pointers are:";
        for (int j = 0; j<5; j++)
        {
                cout<<*(p+j) ;
        }
}
```

Output is : Enter elements of array: 1 2 3 4
Array elements displayed using pointers are: 1 2 3 4

C++'s Dynamic Allocation:

C++ provides the way to allocate memory dynamically (at the run time) by 2 memory management operators namely, **new and delete.**

The **new** operator returns a pointer to allocated memory.

Syntax:-- ***ptr_var* = new *datatype* (*initializer*);**

You can initialize the allocated memory to some known value by using an initializer.

The **delete** operator frees memory.

Advantages:

- Size is computed automatically.
- No need for explicit type cast.
- **new** and **delete** can be overloaded.

Warning:--In C++ it is possible to access and modify the unallocated memory locations, but it should be avoided.

E.g.
```
int a[5];
int * p=a[4];   //p points to the last element.
++p;            //now p points to memory past last element.
*p==3;          //danger!!!!!!!!!Don't even think of doing it!!!!!!!!!!!
```

27. Following program demonstrates use of Dynamic Memory Allocation:

```
#include <iostream.h>
#include <stdlib.h>
void main()
{
        int *p;
        p = new int (50); // initialize to 50
        if (!p)
        {
                cout << "Allocation error\n";
                exit(1);
        }
        cout << "At " << p << " ";
        cout << "the value is " << *p << "\n";
        delete p;
}
```

Output is : At 0x00780DA0 the value is 50

Functions, returning the reference:-

A function may return a reference. A function's return type may be a reference provided that the value returned is a lvalue, which is not local to the function, which gives us a facility for treating function as lvalue (on the left side of the assignment of operator)

28. Following program demonstrates returning refernce from function.

```
int & returnref(int &,int &);
#include<iostream.h>
void main()
{
        int a=2,b=1;
       cout<<a<<b<< returnref (a,b);
       returnref (a,b)=5;
       cout<<a<<b<< returnref (a,b);
}
int & returnref(int &a,int &b)
{
        if(a>b)
                return a;
       else
                return b;
}
```

Output is : 2 1 2
5 1 5.

Thus returnref() function allows us to return a reference to the larger of two variables passed to it.

QUESTIONS

1 Fill in the blanks by selecting correct answer.

i) In _____control structure, depending on a certain condition a certain series of events will be executed. (iterative, branching)

ii) In _____ technique changes made in function are local to the function and not reflected in caller function. (call by value, call by reference)

iii) Conditional operator is equivalent to following control structure.
(for, do ... while, if ...else)

iv) In _____ loop, condition is tested after first iteration. (do ... while, while)

v) _____ is used to continue next iteration of loop. (break, continue, goto)

vi) _____ is unconditional control structure. (goto, if...else, switch-case)
vii) _____ is called as circular definition. (Inline function, Recursive function)
viii) Strings in C++ are treated as _____ (integer array, float array, character array)
ix) _____ is an alias for the variable to which it is assigned. (reference, pointer)
x) In context with the pointers, _____ is the de-reference operator. (&, *)

Answer: i) branching ii) call by value iii)if...else iv)do...while v) continue vi) goto vii) Recursive function viii) character array ix) reference x) *

2. Answer the following.

i) What are control structures in C++ ?

ii) Explain the difference in while and do...while loops

iii) Explain else-if ladder with the example.

iv) Explain the significance of break in switch-case construct.

v) What is infinite loop? How it can be broken?

vi) Explain the use of continue statement in C++.

vii) Can program control be passed unconditionally, if yes how?

viii) Is it possible to have nesting of the loops in C++? How?

ix) What is function? Explain advantages of using function in your program.

x) What is the difference between function prototype, function call and function definition?

xi) Explain following terms associated with the functions:

a).Recursive function .b)Default arguments. c) Function overloading

d)Pass by value Vs. Pass by reference. e) Inline functions

xii) What is array? Explain advantages of arrays.

xiii) Explain multidimensional arrays.

xiv) What are pointers and reference in C++?

xv) How array can be traversed with the pointers?

xvi) Explain how arrays are passed to the function.

Programs for Practice

1. Write the program to do following using for and do—while loop.

   1+3+5+7+.....n the value of n should be accepted from user.

2. Write a program to display Fibonacci series using recursion.

3. Write a program for 4 function calculator using if-else and switch case.

   (Here accept the 2 numbers (operands) from the user; also accept type of operation to be performed on these operands. If user inputs "*", it should display multiplication of given numbers, If user inputs "/", it should display division of given numbers etc).

4. Take the year as an input from user and decide whether it is leap year or not.

5. Display following using for loop.

   a)
   ```
   *******
   *****
   ***
   *
   ```

   b)1
   ```
   1 2 3
   1 2 3 4 5
   1 2 3 4 5 6 7
   ```

   c)
   ```
   ***********
    *******
      ***
       *
   ```

6. Display following

   ```
   1
   1 2 3
   1 2 3 4
   1 2 3 4 5
   ```

   (Here number of rows should be accepted from user.)

7. Using while loop compute sum of the series: $1^2 + 2^2 + 3^2 + ...+n^2$

   (The value of n should be accepted from user.)

8. Using multidimensional arrays display addition of 2 matrices.

❑❑❑

4.VISUAL BASIC

INTRODUCTION

This chapter introduces VB's Integrated Development environment (IDE) and the basic principles of developing applications with visual tools and event programming. We will study the steps of creating a simple application (Math Calculations) and the component of the Visual Development Environment.

In this chapter, we will take a tour of the environment and discuss the basics of developing user interface. After completing this chapter one can write a program in VB with few lines.

4.1 ONE LANGUAGE, THREE EDITIONS

We assume that you have installed VB6. This means you have already decoded which edition of VB you are going to use.

Visual Basic comes in three editions

1) The VB Learning Edition
2) The VB Professional Edition
3) The VB Enterprise Edition

The Visual Basic Learning Edition

It is the introductory edition that lets you easily create windows applications. It covers with all the tools you need to build mainstream windows applications.

The Visual Basic Professional Edition

It is for computer professionals and includes advanced features such as tools to develop Active X and Internet Controls.

The Visual Basic Enterprise Edition

It is the advanced edition and is assigned at programmers who build distributed applications in a team environment. It includes all the features of the Professional Edition, plus tools such as Visual Source Safe (a version control system) and the Automation and Component Manager.

Depending on which edition of Visual Basic you have installed, you may see fewer or more options in the menus and toolbars of the Visual Basic IDE.

VB is not just a language. It is an Integrated Development Environment in which you can develop, run, test and debug your applications.

4.2 Study Of Integrated Development Environment (IDE)

Getting started with Vb – Using the IDE

With VB, you can create the following types of applications:

a) **Standard EXE:** It is a typical application.

b) **Active X EXE, Active X DLL:** These types of projects are available with the professional edition. Active X components are OLE automation services. Active X components are basic code – building components that don't have a visible interface and can add special functionality to your applications.

c) **Active X Control:** An Active X Control such as a text box or a command button control is a basic element of the user interface. You can build your own custom controls.

d) **Active X Document EXE, Active X Document DLL:** Active X documents are VB applications that run in the environment of a container that supports hyperlinking. (such as Internet Explorer)

e) **VB Application Wizard, VB Wizard Manager:** The application wizard takes you through the steps of setting up the skeleton of a new application.

The wizard manager lets you build your own wizard. A wizard is a sequence of windows that collect information from the user.

f) **Add-In:** You can create your own add-ins for the VB IDE. Add-ins are special commands you can add to VB's menus (they appear under the Add-Ins menu)

The "New Project" window has three tabs: New, Existing and Recent. In the New tab, you can select the type of a new project, as explained already.

Existing tab will select an existing project and open it.

Recent tab is clicked to open a project you worked on recently. It contains a list of the most recently opened projects.

Select the "Standard EXE" icon in the new project window and then click OK to open a new VB project.

What you now see on your screen is the VB IDE, which is made up of a number of components. The main window, in the center contains a form (Form 1). The form is the application's window, and in it you place the element of your applications user interface (check boxes, command buttons and so on).

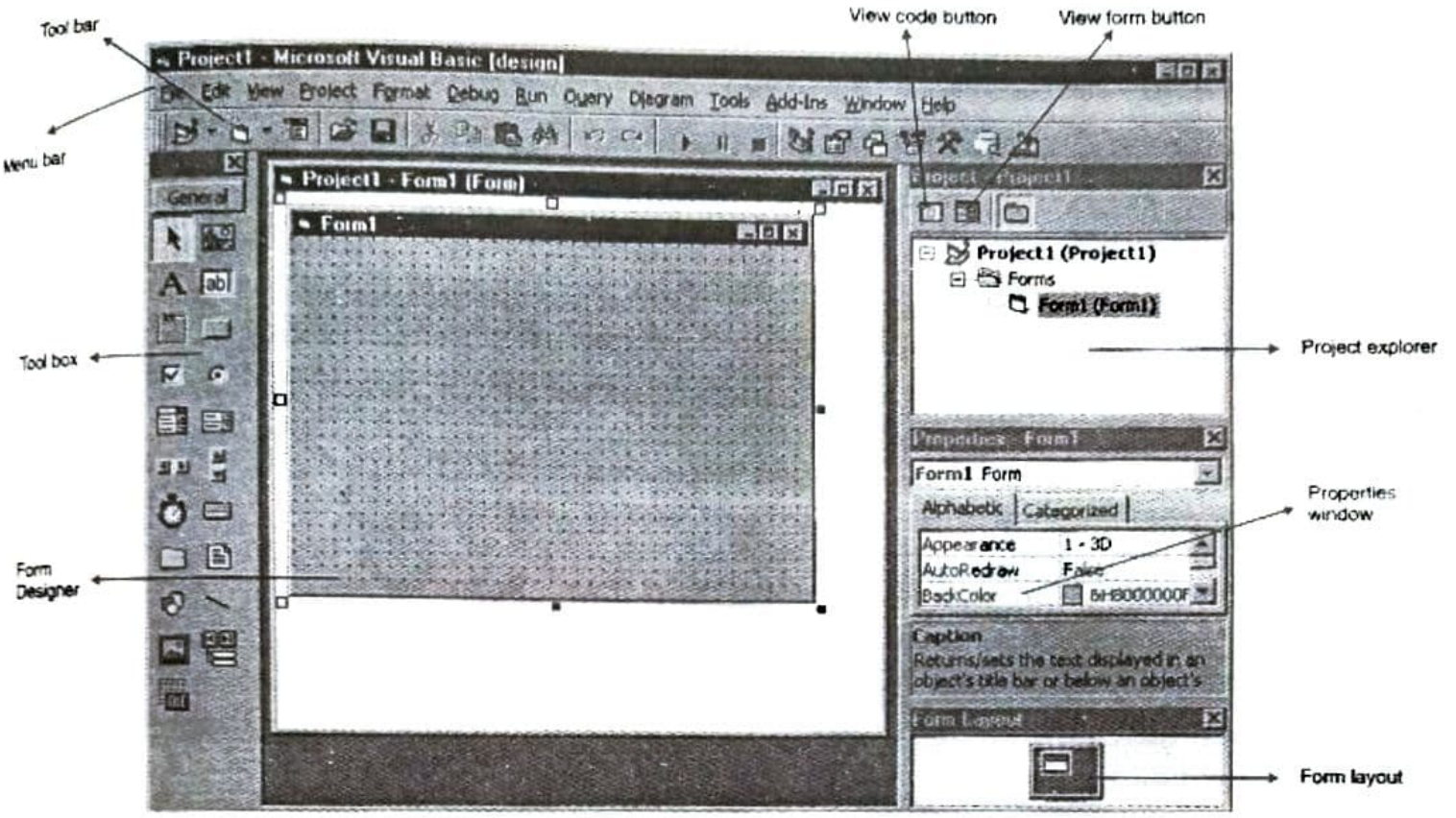


Fig.(4.1) A new Visual Basic Project Screen

The Menu Bar

The basic menus are:

a) **File:** Contains commands for opening and saving projects and creating executable files and a list of recent projects.

b) **Edit:** Contains editing commands. e.g. undo, copy, paste, find, replace.

c) **View:** Contains commands for showing or hiding components of the IDE.

d) **Project:** Contains commands that add components to the current project, references and new tools to the toolbox.

e) **Format:** Contains commands for aligning the controls on the form.

f) **Debug:** Contains usual debugging commands.

g) **Run:** Start, break and end execution of the current application.

h) **Query:** Available when building database applications.

i) **Diagram:** Available when building database applications used for editing database diagrams.

j) **Tools:** Contains tools those are needed in building Active X components and Active X controls.

k) **Add-Ins:** Command, to add and remove add-ins.

l) **Window:** Commands to arrange windows on the screen.

m) **Help:** Contains information to help you as you work.

The Project Explorer

Displays the components of the project. simple projects are made of a single form. The project window is called Project Explorer, because it has the look of Window Explorer.

Toolbox and Property Editor

The Toolbox

Contains the icons of the control you can place on a form to create the application's user interface. By default, the toolbox contains the pointer icon and the icons of 20 Active X controls. To place a control on a form, you first select it with the mouse and then move the mouse over the form. The cursor turns into a cross and you can draw the control on the form.

In addition to the default toolbox (called general) you can create custom layouts by right-clicking the toolbox and selecting "Add tab" from the shortent menu.

The Properties Window

Contains the property settings for the selected control. Properties are attributes of an object such as, its size, caption and colour. For e.g. caption property for a command button, backcolour property of form, appearance (0 for **flat look** and 1 for 3 dimensional) etc.

The Form Designer

It is the main window in the middle of the screen and in it you design and edit the application's user interface. The same window displays text editors in which you can enter and edit the applications code. The form designer displays two windows for each form: a)The form itself, b) A code window

The Form Layout

You can use the form layout window, in the lower-right corner of the VB IDE, to determine the initial positions of the forms in your application. This window is useful in application's that use multiple forms because you can specify how each form is positioned with respect to the main form.

The Immediate Window

At the bottom of the IDE is a debugging aid. You can step into the applications code while it executes, change the values of the variable or even insert statements in the code, and then continue the execution of an application.

4.3 The Elements Of The User Interface - Use Of Controls

The first element of the user interface is the Form. This is the window displayed at runtime, and it acts as a container for all the elements of the interface. The elements are shown as icons in the toolbox.

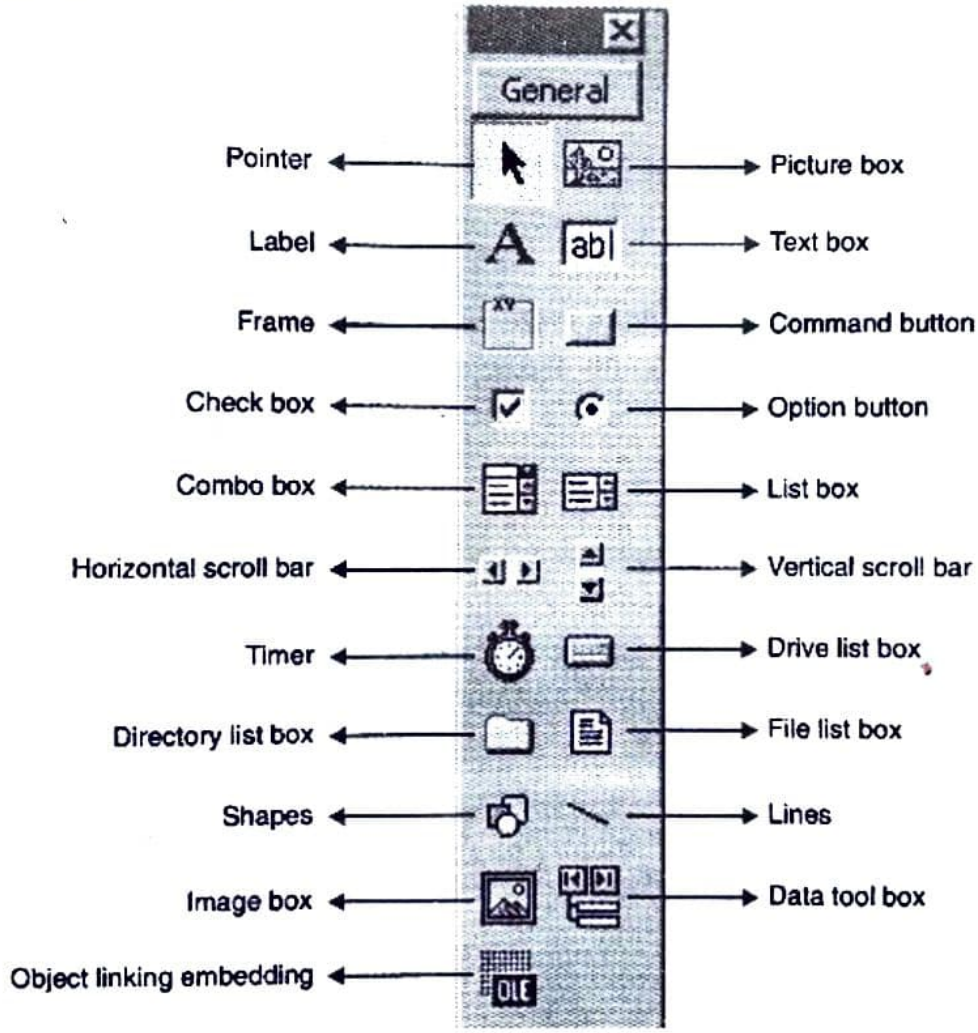


Fig. (4.2) Layout of toolbox

a) **Picture Box:** This control is used to display images, and the images are set with the picture property.

b) **Label:** This control displays text on a form that the user can't edit. Set the labels text with the "Caption" property.

c) **Textbox:** Displays text that the user can edit. It is a mini text editor. "Text" property can set the text or read the text that the user enters.

d) **Frame:** Used to draw boxes on the form and to group other elements.

e) **Command Button:** Represents an "action" i.e., carried out when the user clicks the button.

f) **Check Box:** Presents 1/more choices that the user can select. Main property is "Value", and it is '0' if the check box is cleared, and '1' if the check box is checked.

g) **Option Button (or Radio buttons):** Appear in groups and the user can choose only 1 of them. Main property is "checked" and it is 'True' if the control is checked, and 'False' otherwise.

h) **Combo Box:** Similar to the List Box control, but it contains text edit field. The user can enter/choose an item from the list and enter a new string in the edit field.

i) **List Box:** Contains a list of options from which the user can select one or more. The user can scroll the list to locate an item, unlike a group of check boxes or option buttons.

j) **Horizontal and Vertical Scrollbar:** It let's the user specify a magnitude by scrolling the control's button between its minimum and maximum value.

k) **Timer:** To perform tasks at regular intervals. Main property is "Interval" which determines how often the timer notifies your application.

l) **Shape:** Used to draw graphical elements.

m) **Line:** Similar to shape control, the line control is used to draw lines on the form.

n) **Image:** similar to the picture box control, in that it can display images, but supports few features of the picture box control.

o) **Data:** Provides point-and-click access to data stored in databases.

p) **OLE:** This control is a window you can place on your form to host documents from other applications, such as MS Word or Excel.

The property setting

The properties Window shows the property setting menus like caption, control Box, Font as shown in fig.(4.3). With these settings we can arrange the appearance of the controls on the form.

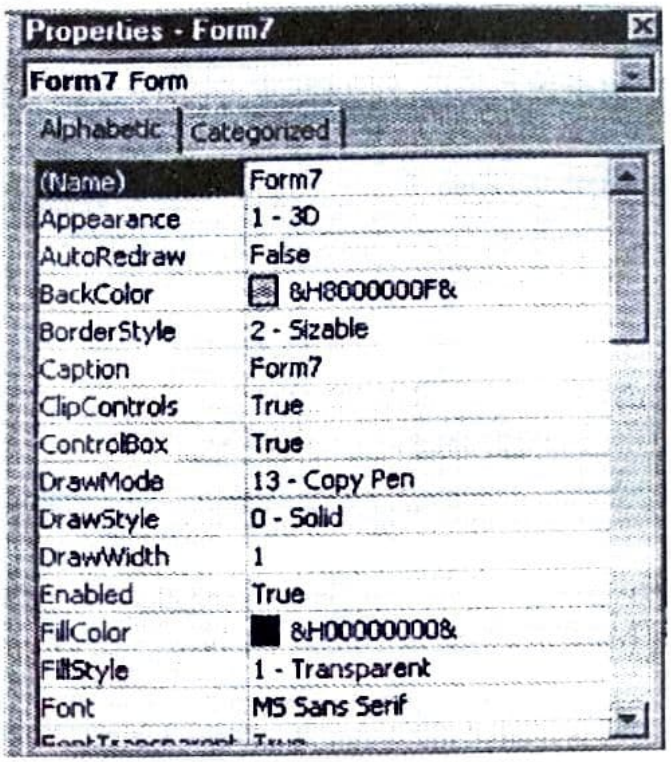


Fig. (4.3) Property Window

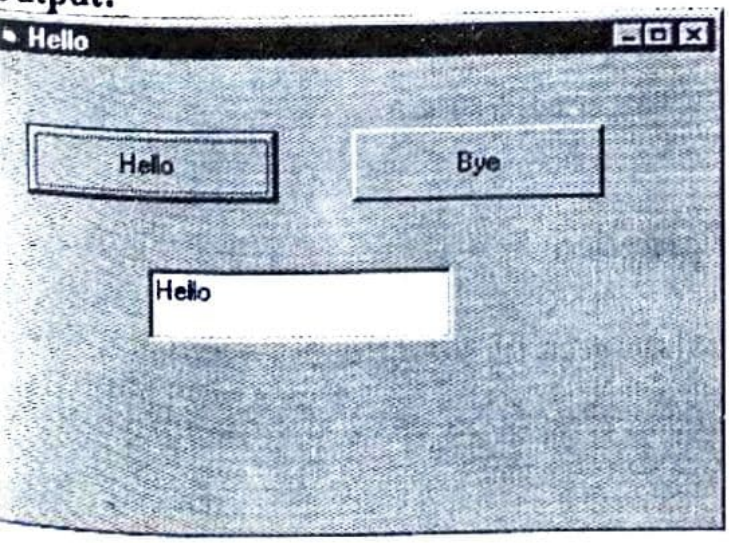
Let us see a very simple window design:

Program 1: Program to display "Hello"

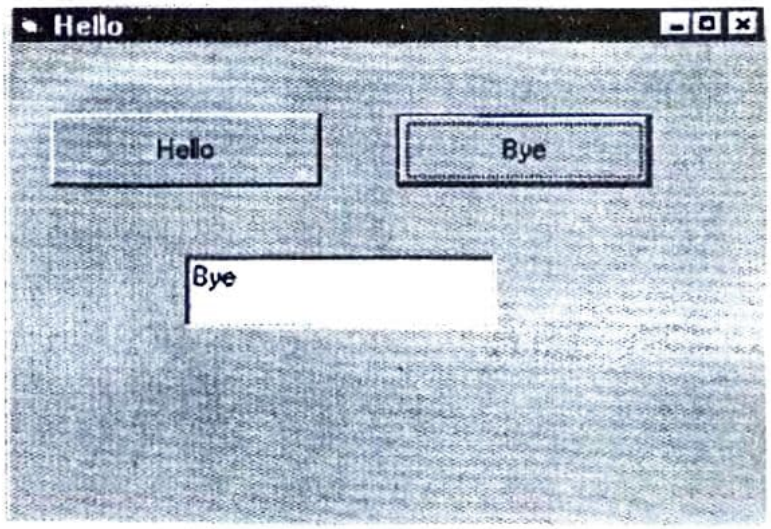
Code:

```
Private Sub Command1_Click()
Text1.Text = "Hello"
End Sub

Private Sub Command2_Click()
Text1.Text = "Bye"
End Sub
```

Output:



After pressing Hello



After pressing Bye

Program explanation: Command buttons

1) Command1 : caption = Hello, On click event of this button Hello is displayed in text box.
2) Command2 : caption = Bye, On click event of this button Bye is displayed in text box.

QUESTIONS

Q. 1) Select the correct alternative and rewrite the following:

1) _____ is the extension for Visual Basic project file.
a) vbp, b) exe, c) vbs, d) Vba

2) _____ control appears in group and user can choose only one of them.
a) Check box b) Option button c) Command d) None of these

3) _____ control displays list of all folder on current driver.
a) Drive list box b) Directory list box c) File list d) None of these

4) _____ control is used to insert document from another application.
a) OLE b) Shape c) OEL d) None of these

Answers: 1) vbp 2) option button 3) Directory list box 4) OLE

2. State the features of Visual Basic.

3. What are various components of Visual Basic IDE? Explain it.

4. State use of following controls in Visual Basic.
a) Text box b) Command button c) Combo box d) Frame

5. Explain use of following window.
a) Form layout, b) Properties window

Programs for practice

1) Create an application in Visual Basic to calculate area of triangle.
2) Create application in Visual Basic to calculate simple interest if principle amount, period, and rate is given.

4.4 VB PROGRAMMING

1) **Variables:** In VB, variables store values during a program's execution. Variables are placeholders in whom you can leave values and recall them at will. A variable has a name and value. In your code, you can refer to the value of a variable by the variable's name.

2) **Declaring Variable:** Variables must be declared in advance for the compilers. If the compiler knows all the variables and the types that are going to be used in the application ahead of time, it can produce an optimized code.

Explicit Declarations: to declare a variable, use the "Dim" statement, followed by the variable name and type as follows:

```
Dim meters as Integer

Dim greetings as String
```

Implicit Declarations: The variable type is variant, the generic data type that can accommodate all other data types VB adjusts its type according to the value you assign to it.

```
Dim var1, var2
var1 = "Thank you"
var2 = 49.99
```

The var 1 variable is a string variable and var 2 is a numeric one.

Variable Naming Conventions

A variable name:

1) Must begin with a letter.
2) Can't contain an embedded period or any of the type – declarations character.
3) Must not exceed 255 characters.
4) Must be unique within its scope.

Types of Variables

1) Numeric
2) String
3) Boolean
4) Date
5) Object
6) Variant

There are a variety of numeric data types, viz.

1) Integer
2) Single
3) Double
4) Long
5) Currency

Length Strings

String variables have variable lengths, you can also specify fixed length strings with a statement, like;

Dim some text as string * 1000, the variable is long enough to hold 1000 characters.

Date Variables

Dim day1 as Date, day2 as Date,

Day1 = Date ( )

The day1 variable will be the current date, you can easily find the number of years, months and days between 2 dates with the help of year ( ), month and day ( ) functions.

e.g.: Year (day 2 – day 1)

Month (day 2 – day 1)

day (day 2 – day 1)

Object Variables

e.g. Dim a as command button,
b as command button

set a = command 1

set b = command 2

a·caption = "Hi"

b·caption = True

3) **Constants:** Some variables don't change the value during the execution of a program. There are "constants" that appears many times in your code. For instance, if your program does math calculations, the value of pi (3.14159...) may appear many times in your code. These values are best represented by constants.
   a) Constants don't change value
   b) Constants are produced faster than variables

   Declaration: Constant constantnance [as type] = value

   e.g. Public constant pi as Double = 3.14.

   The constant name follows same rules as variable names.

4) **Arrays:** A standard structure for storing data in any Programming Language is an array. Arrays can hold sets of related data. The values stored in an array can be accessed by an index.

Declaring Arrays

Dim Salary (15) the parentheses include the maximum numbers of elements the array can hold.

Salary (0) is 1st person's salary, Salary (1) is 2nd person's salary, and so on.

Dim names (15)

Assign values to elements of both arrays, as follows:

names (0) = "for doe"

salary (0) = 34000

.

.

This structure is more compact and convenient than having to hard code. The names and salaries of employees in variables.

You can specify the type of elements with the "As" keyword.

Dim names (15) As String

Dim salary (15) As Long

All elements in an array have the same data type. But, when the data type is variant, the individual elements can contain different kinds of data.

Multidimensional Arrays

A 2-dimensional array has 2 indices. The first identifies the row and the second identifies the column. The benefit of using multidimensional arrays is that they are conceptually easier to manage.

e.g. an array board could be declared as follows:

Dim board (10,10) As Integer

Board (1,1) = 0

Dynamic Arrays

Sometimes you may not know how large to make an array. Instead of making it large enough to hold the maximum number of data, you can declare a dynamic array. The size of a dynamic array can vary during the course of the program. To create a dynamic array declare it as usual with the Dim statement, but not specify its dimensions

Dim Dym Array( )

Later in the program, when you know how many elements you want to store in the array, use the ReDim statement to redimension the array with its actual size.

Redim DymArray (10)

5) **Functions:** A function is similar to a subroutine, but a function returns a result. Because a function reports a result, it must have a type.

e.g. Date ( ) => returns current date

Abs ( ) => returns absolute value of its argument. if the argument is +ve, it returns it as it is, if it is –ve, it inverts its sign. Vb and all other PL provide a number of built-in functions for those needed most frequently by developers. But each developer has special needs and you can't expect to find all the procedures you may ever need in a PL. Sooner or later, you will have to supply your own. Functions are called by name, and a list of arguments follows the name in parentheses. If you don't care about the return value, you can call the function with the call statement,

e.g. Call Convert ( )

Built-in functions

Function	Return Value
Abs	Absolute Value of a number
Asc	ASCII or ANSI code of a character
Chr	Character corresponding to given ASCII or ANSI code
Cos	Cosine of an angle
Date	Current date as a text string
Now	Current time and date
Sin	Sine of an angle
Sqr	Square root of a number
Str	Number converted to a text string
Time	Current time as a text string
Val	Numeric Value of a given string

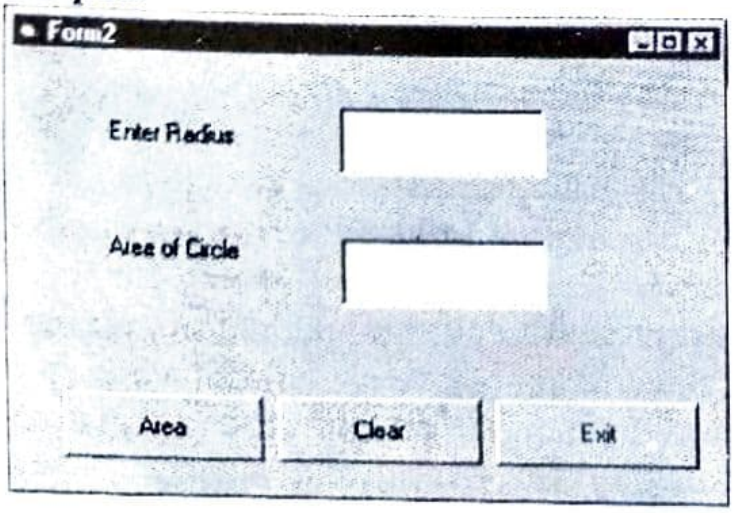
Program1: To find the Area Of Circle

Code:

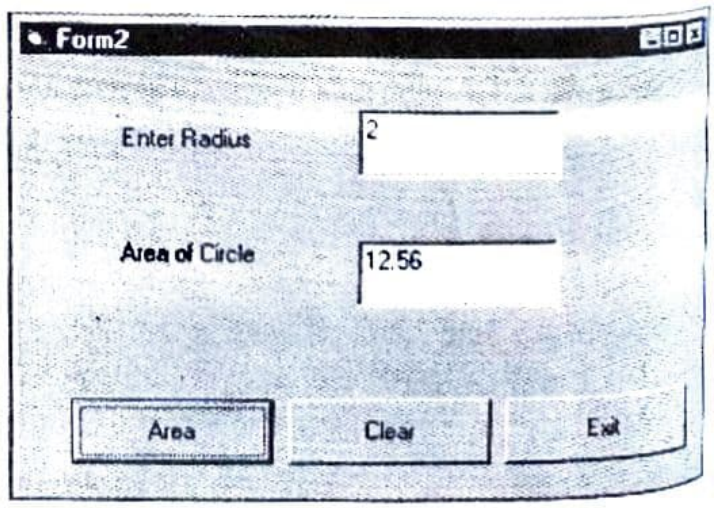
```
Private Sub Command1_Click()
Text2.Text = 3.14 * Text1.Text * Text1.Text
End Sub

Private Sub Command2_Click()
Text1.Text = " "
Text2.Text = " "
End Sub

Private Sub Command3_Click()
End Sub
End
```

Sample Programs

Output:



After executing the Program



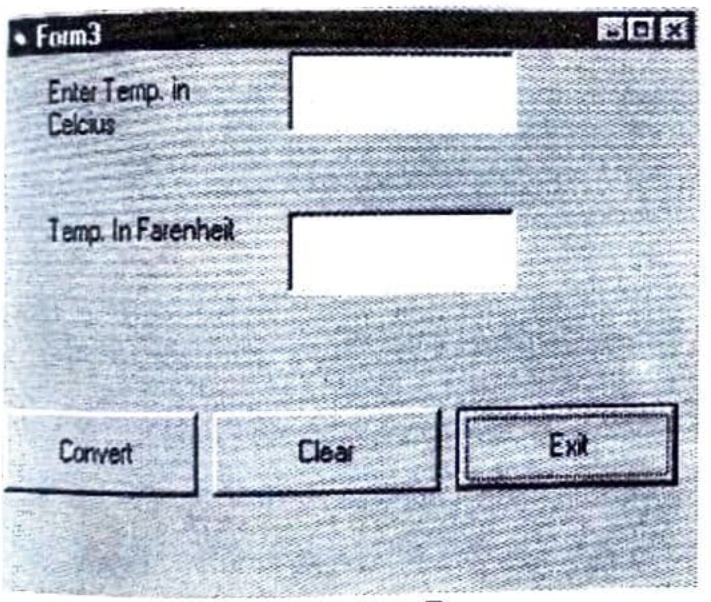
After entering the value

Program2: Conversion of temperature

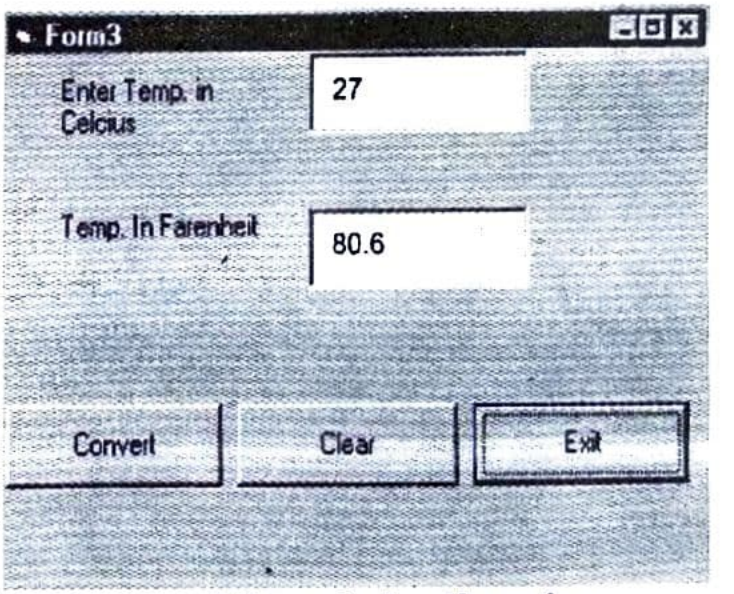
Code:

```
Private Sub Command1_Click()
Text2.Text = 1.8 * Text1.Text + 32
End Sub

Private Sub Command2_Click()
Text1.Text = ""
Text2.Text = ""
End Sub

Private Sub Command3_Click()
End Sub
End
```

Output:



After executing the Program



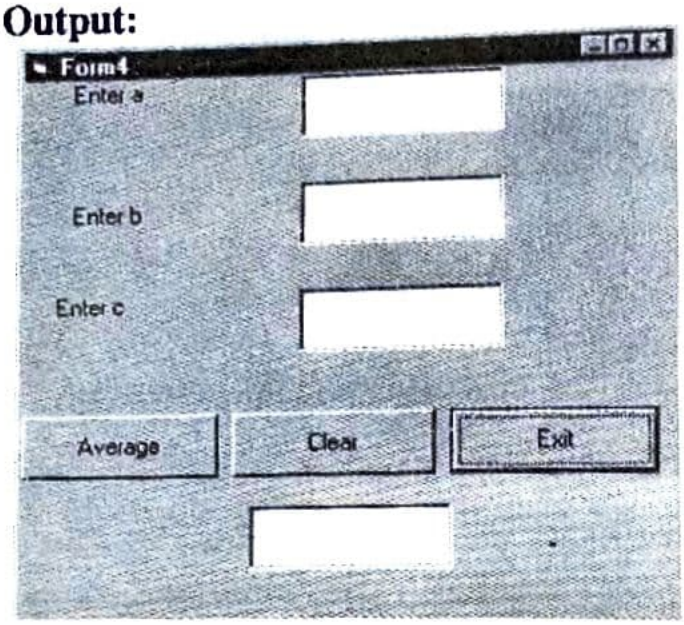
After entering the value

Program3:To find average of 3 numbers.

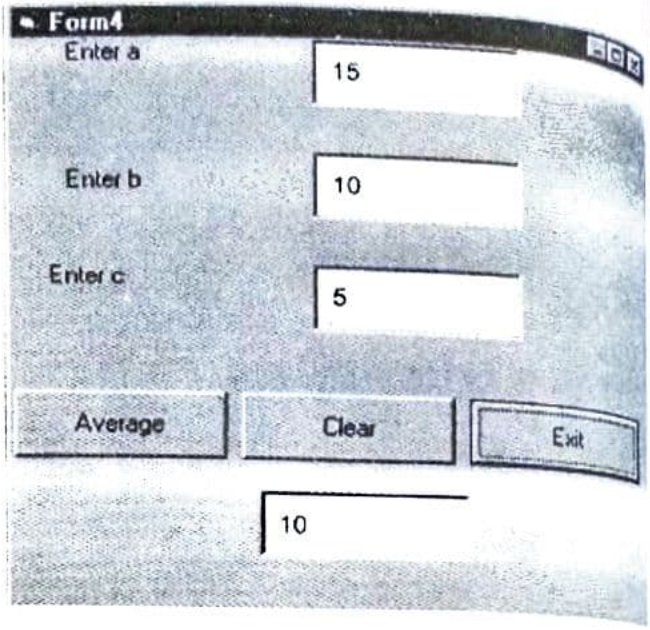
Code:

```
Private Sub Command1_Click()
Text4.Text = Str((Val(Text1.Text) + Val(Text2.Text) + Val(Text3.Text)) / 3)
End Sub
Private Sub Command2_Click()
End Sub
End
```

Output:



After executing the Program



After entering the value

6) **Control Flow Statements:** VB provides three control flow, or decision structures.
   a) If ..... Then
   b) If ..... Then ..... Else
   c) Select Case

a) **If ..... Then:** The If.....Then structure sets the condition specified, and if it is true executes the statement that follow. The If structures "Can have a single-line or multiple-line syntax".

Single-line syntax

If condition Then statement

VB evaluates the condition, and if it is True, executes the statement that follows. If the condition is False, it continues with the statement following the If structure.

You can also execute multiple statements by separating them with a colour.

Multiple-line syntax

If condition Then statement

If Month (date) = 1 then year = year + 1

Break it into multiple lines, as:

```
if   month (date) = 1 then
     year = year + 1
end if
```

b) **If.....Then.....Else:**

Syntax:
```
if      condition then Statement block – 1
else
        statement block – 2
end if
```

Else or Else if keyword

```
if      condition 1 then statement block – 1
else if condition 2 then statement block – 2
else
        statement block – 3
end if
```

c) **Select Case:** The select case structure tests a single expression, which is evaluated once at the top of the structure. The result of the test is then compared with several values, and if it matches with one of them then, the corresponding block of statements is executed.

Select Case Expression

```
Case  Value 1
      Statement block – 1
Case  Value 2
      Statement block – 2
.
.
Case Else
      Statement block
End Select
```

7) **Loop Statements:** Allow you to execute one or more lines of code repetitively VB supports all the loop statements.

a) Do.....Loop
b) For.....Next
c) While.....Wend

a) **Do.....Loop:** Executes a block of statements as long as a condition is True. There are two variations of the Do.....Loop statement. A loop can be executed either "while the condition is True" or "until the condition becomes True". these two variations use the keywords "while" and "until" to specify how long the statements will be executed.

Syntax

1) Do While Condition
   Statement-block
   Loop

OR

2) Do Until Condition
   Statement-block
   Loop

Another variation of the Do loop executes the statements first and evaluates the condition after each execution.

Syntax

1) Do
   Statements
   Loop while condition

OR

2) Do
   Statements
   Loop until condition

b) **For.....Next:** Unlike the Do....Loop the For.....Next requires that how many times the statements in the loop will be executed. For.....Next loop uses a counter that increases / decreases in value during each repetition of the loop.

Syntax

```
For Counter = Start to end
        [ Step increment]
Next [counter]
```

The counter variable need not be listed after the next statement, but it makes the code easier to read, especially when For.....Next loops are nested within each other.

c) **While.....Wend:** The While.....Wend loop executes a block of statements while a condition is true.

Syntax

```
While condition
        statement-block
Wend
```

8) **Nested Control Structures:** You can place, or nest, control structures inside other control structures. Control structures in VB can be nested in as many levels as you want. Here is the structure of a nested For.....Next loop that scans all the elements of a 2-dimensional array.

```
For irow = 0 to unbound (Array 2 (0))
For icol = 0 to unbound (Array 2 (1))
        {process element Array 2 (i row, i col)}
        Next icol
Next irow
```

You can also nest control flow structures, the following structures tests a user supplied value to determine if its +ve and, if so, determine whether the value exceeds a certain limit.

```
Income = Input Box ("Enter your income")
If Income >0 Then
   If Income > 20000 Then
         Msg Box "Pay taxes"
   else
         Msg Box "No tax"
   end if
else
   Msg Box "?"
end if
```

The Exit Statement

Allows you to exit permanently from a block of statements in a control structure, from a loop, or even from a procedure.

e.g.:

```
For i = 0 To unbound (nArray ( ))
   If nArray (i) < 0 Then Exit        or
   nArray (i) = Sqr (nArray(i))
Next
```

If a (–)ve element is found in this loop, the program exits the loop and continues with the statement following the Next keyword.

Similarly, these are "ExitDo", "Exit Function" and "Exit Subroutine" statements.

e.g.:

```
For i = 0 To unbound (nArray ( ))
         If nArray (i) < 0 Then Exit Function
         nArray (i) = Sqr (nArray (i))
Next
```

Exits not only the loop, but also the function itself.

9) **Event-Driven Programming:** Forms and controls are the basic elements in the user interface of any windows application. In VB, these elements are called "Objects", because they are manipulated like objects. Objects have properties and methods and they react to external events, as does any physical object.

A Few Common Properties

Following apply to most objects:

a) Name b) Appearance c) Back Colors

d) Fore Color e) Font f) Caption
g) Text h) Width, Height i) Left, Top
j) Enabled k) Visible

4.5 A FEW COMMON METHODS

Objects have methods. You can think of methods as the actions of an object.

a) **Clear:** tells the control to discard its contents.

b) **Move:** Lets you move and resize controls from within your application's code.

c) **AddItem, RemoveItem:** Used to manipulate the items in a List Box or a Combo Box control.

A Few Common Events

Events determine the control's reactions to external conditions. Controls recognize events, but your application handles them. A command button will recognize that it was clicked, but it won't react to the event, unless you provide some code. Once you specify a subroutine for the control's Click event, this subroutine executes each time the control is clicked. The subroutine that determines how a control reacts to an event is called event handlers.

The combination of the control's name and the event's name is unique and is the name of the event handlers. Each time an event takes place, VB looks for the subroutine made up of the name of the control on which the event took place and the name of the event. If such a handler exists, it's executed. If not, your application won't react to the event.

Mouse Events: Events triggered by mouse actions.

b) **Click, Double Click:** The click event takes place when the user clicks the left mouse button. The double click events take place when the user double clicks the left mouse button.

c) **MouseDown, MouseUp:** Mouse Down event takes place when the mouse button is pressed. The Mouse Up event takes place as it is released.

d) **Mouse Move:** Takes place continuously as the mouse is moved over a control.

Keyboard Events: Generated by keystrokes. Usually you must program the keyboard events for the controls that can accept text.

a) **KeyDown, KeyUp:** Key Down event is triggered when a key is pressed and Key Up when a key is released.

b) **Key Press:** To find out which key is pressed? This event is used frequently to write keyboard handlers for text boxes, because this event takes place before the

character pressed is displayed in the text box. The "Key ASCII" arrangement is the ASCII character of the key pressed.

c) **Change:** Triggered by various controls when their contents change.

d) **Got Focus and Lost Focus:** When the focus is moved from one control to another, the 1st control receives the lost focus event, and after that, the 2nd control receives the got focus event. You can use the got focus event to initialize a control for editing, and you can use the lost focus event to validate the data entered by the user.

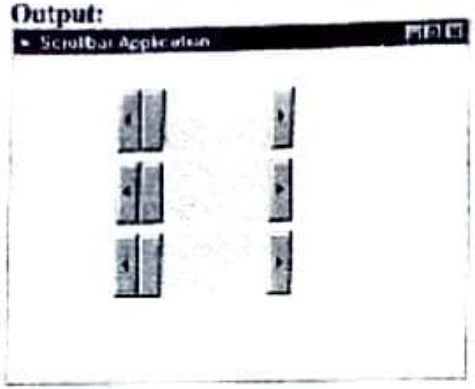
SIMPLE PROGRAMS

Program1: Scrollbar Application

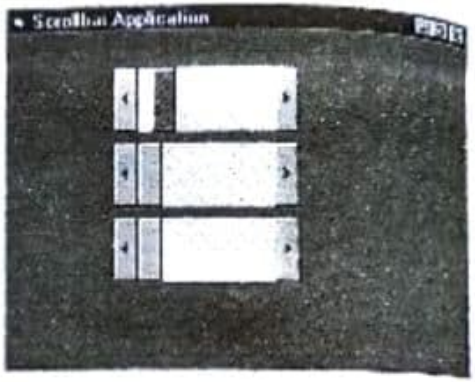
Code:

```
Private Sub HScroll1_Change()
Form6.BackColor = RGB(HScroll1.Value, HScroll2.Value, HScroll3.Value)
End Sub

Private Sub HScroll1_Scroll()
Form6.BackColor = RGB(HScroll1.Value, HScroll2.Value, HScroll3.Value)
End Sub

Private Sub HScroll2_Change()
Form6.BackColor = RGB(HScroll1.Value, HScroll2.Value, HScroll3.Value)
End Sub

Private Sub HScroll2_Scroll()
Form6.BackColor = RGB(HScroll1.Value, HScroll2.Value, HScroll3.Value)
End Sub

Private Sub HScroll3_Change()
Form6.BackColor = RGB(HScroll1.Value, HScroll2.Value, HScroll3.Value)
End Sub

Private Sub HScroll3_Scroll()
Form6.BackColor = RGB(HScroll1.Value, HScroll2.Value, HScroll3.Value)
End Sub
```

Output:



After executing the Program



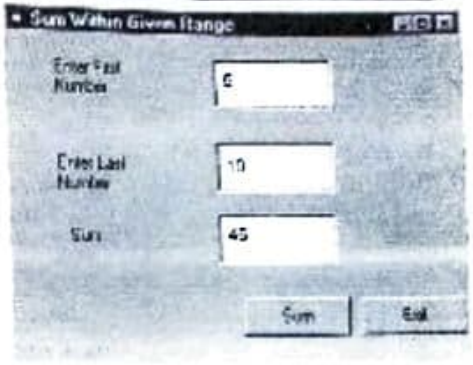
After scrolling

Program2: Sum of numbers within a range (using for /while)

Code:

```
Private Sub Command1_Click()
Dim sum As Double
Dim k As Integer
sum = 0
i = Text1.Text
j = Text2.Text
Using For Loop the code will be
'For k = i To j
'sum = sum + k
'Next k
Using while Loop the code will be
k=i
While k <= j
sum = sum + k
k = k + 1
Wend

Text3.Text = sum

End Sub
```

After executing the Program



After entering the value

Program 4:A Design of Simple Calculator (Using select case)

Code:

```
Option Explicit
Dim x As Double
Dim operand1 As Double, operand2 As Double
Dim clear display As Boolean
Dim operator As String

Private Sub Command1_Click()
Text1.Text = Text1.Text + "1"
End Sub
```

```
Private Sub Command10_Click()
Text1.Text = ""
End Sub

Private Sub Command11_Click()
Text1.Text = Text1.Text + "0"
End Sub

Private Sub Command12_Click()
Text1.Text = Text1.Text + "."
End Sub

Private Sub Command13_Click()
operand1 = Val(Text1.Text)
operator = "+"
Text1.Text = ""
End Sub

Private Sub Command14_Click()
operand1 = Val(Text1.Text)
operator = "-"
Text1.Text = ""
End Sub

Private Sub Command15_Click()
operand1 = Val(Text1.Text)
operator = "*"
Text1.Text = ""
End Sub

Private Sub Command16_Click()
operand1 = Val(Text1.Text)
operator = "/"
Text1.Text = ""
End Sub
Private Sub Command17_Click()
Dim result As Double
Operand2 = Val(Text1.Text)
If operator = "-" Then result = operand1 - operand2
If operator = "*" Then result = operand1 * operand2
```



```
If operator = "+" Then result = operand1 + operand2
If operator = "/" And operand2 <> "0" Then result = operand1 / operand2
Text1.Text = result
End Sub

Private Sub Command18_Click()
Text1.Text = Text1.Text * -1
End Sub

Private Sub Command2_Click()
Text1.Text = Text1.Text + "2"
End Sub

Private Sub Command3_Click()
Text1.Text = Text1.Text + "3"
End Sub

Private Sub Command4_Click()
Text1.Text = Text1.Text + "4"
End Sub

Private Sub Command5_Click()
Text1.Text = Text1.Text + "5"
End Sub

Private Sub Command6_Click()
Text1.Text = Text1.Text + "6"
End Sub

Private Sub Command7_Click()
Text1.Text = Text1.Text + "7"
End Sub

Private Sub Command8_Click()
Text1.Text = Text1.Text + "8"
End Sub

Private Sub Command9_Click()
Text1.Text = Text1.Text + "9"
End Sub
```

QUESTIONS

1. Select the correct alternative and rewrite the following.

1) _____ function displays current date and time.
a) now ( ) b) date ( ) c) day ( ) d) none of these

2) _____ type of variable stores either true or false value.
a) Boolean b) object c) integer d) none of these

3) _____ statement is used to terminate for loop.
a)exit for b) exit c) break d) none of these

4) _____ command is used for aligning the controls
a) Edit b) Format c) Run, d) none of these

5) For flat look, the appearance should be.....
a) 1 b) 2 c) 0, d) none of these

Answer: 1) now, 2) Boolean, 3) exit for 4) Format 5) 0

2. What are data types available in Visual Basic? Explain it.

3. What is difference between implicit declaration and explicit declaration of avariable in Visual Basic?

4. What is an array? How is it declared in Visual Basic?

5. Explain Do ..... loop and for ..... next loop with suitable example.

6. What is use of exit statement in Visual Basic?

7. State the use of following functions in Visual Basic.
a) str b) val c) abs

8. Explain if ....., then statement with suitable example.

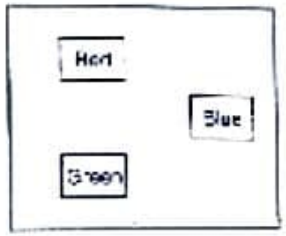
9. What are built-in functions in VB?

10. Enlist different control statements with syntax.

Programs For Practice

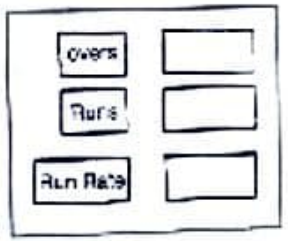
1. Create an application in VB to find maximum number from given two numbers

2. Create an application in VB as follows:



According to click event of button it should change forms background colour. (for e.g. if Red button is clicked it should change colour to Red)

3. Design score board window as follows:



It should calculate run rate if overs and runs are given and if overs > 50, it should display proper message.

4. Develop a student's result system, which will accept roll no., name, and marks in all subjects and calculate total and percentage and determine class of the student.

❑❑❑

5.NETWORKING AND INTERNET

INTRODUCTION

The use of computers is becoming an essential part of life. A user works at different places on different computers, then the data to be used may have to be taken on floppy and used at other places. Sometimes it is not possible to use floppies, as the computer does not have floppy drive. Then, it will be difficult to use the data at other place. To solve such problem, if computers are connected to each other through some transmission media, then the information can be transferred from one computer to other. This concept is called as networking.

The computer networks are available everywhere nowadays. So, it is important to understand the concept of computer network and its usage. Along with computer network, Internet also is very popular network. This chapter explains computer networks and Internet concepts and its applications.

5.1 NETWORKING TERMS AND CONCEPTS

Interconnecting a set of computers with each other using transmission media is called a network. With the help of network, the resources and services can be shared. The shared resources can be data, printer, floppy drive, CD drive, fax, modem, etc. The shared services can be a database, a file, a software etc. All interconnected computer must follow a set of communication rules for transmitting and receiving the data. The rules governing computer communication are called protocols.

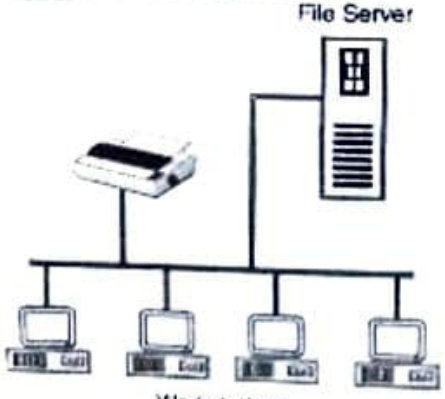


Fig.(5.1)A simple Network

Networking allows us to do the following:

a) Exchanging information
b) Sharing hardware resources
c) Preserving information
d) Protecting information

a) **Exchanging Information:** When computers are connected to each other, then the information of one computer can be transferred to another in a fast and easy way.

b) **Sharing hardware resources:** A network allows the user to share the hardware i.e. printer, floppy drive, CD drive can be shared instead of using separate resources and thus cost of hardware can be reduced.

c) **Preserving information:** The information stored is preserved in a network on a specific storage device. This information can be made available on demand for the authenticated person.

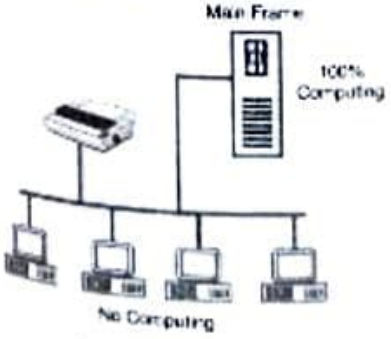
d) **Protecting information:** The information is protected from unauthenticated sources. This way, the privacy is maintained for the stored information i.e. the information will not be made available for the unauthenticated person.

There are three types of network installations:

a) Centralized installation
b) Distributed installation
c) Collaborative installation

a) Centralized Network Installation

These types of networks have a central computer (server) connected to all other computers. The central computer known as mainframe computer does the entire work. The other computers are called as dumb terminals or nodes and work as only input/ output devices.



Centralized Network

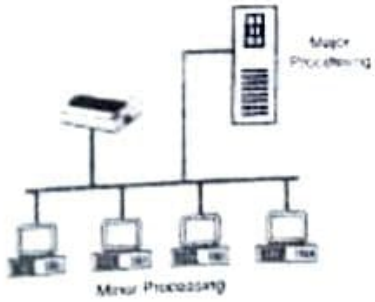


Fig. (5.2) Distributed Network

b) Distributed Network Installation

In a distributed network, the processing is done on each node in individual way. This is because the new PCs (Personal Computers) are with higher processing capacity and higher memory. Using the distributed processing concept, more amount of data gets processed in less amount of time increasing the efficiency.

c) Collaborative Network Installation

This is also called as co-operative processing. This type of processing enables the computer in a collaborative manner to share data, resources and services. The computers work in collaboration with each other i.e. depending upon the available resources to share, the application gets executed in the network in a more better way. An example of Collaborative network is Internet browsing.

5.2 TYPES OF NETWORKS

A Network is an interconnected system that provides communication links among the two or more stations. Each station in a network is known as node. Fig. (5.3) shows a simple communications network with four nodes A, B, C and D each node is connected with other three.

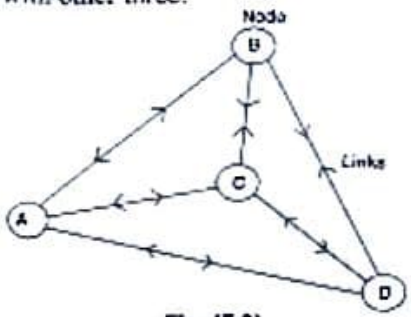


Fig. (5.3)

There are different electronic communication networks, which can be broadly classified according to their coverage area, and their facilities provision, these are :

(a) LAN (Local Area Network)
(b) MAN (Metropolitan Area Network)
(c) WAN (Wide Area Network)

As we know that today there are very common networks which are very popular and we are much familiar with those systems but we can understand their techniques by just comparing one with other as explained below

(a) Local Area Network (LAN)

A local area network is a group of computers connected to each other in a small area. The LANs are situated in a building or a campus.

LAN has following characteristics:

- Data transfer at high rate.
- The geographical area covered by network is limited.
- The resources are connected with different techniques.

A LAN system consists of many PC terminals but a common control terminal is controlling this network known as Server. The Server can be a minicomputer or a mainframe.

(b) Metropolitan Area Network (MAN)

The network smaller than WAN and bigger than LAN is also established hence known as MAN it is a medium size network. An example of MAN is a cable network within a city. Cable network is receiving TV programs from the satellite and it connects many (user). TV sets in the network to watch these programs by such a network containing so many nodes located at different locations in the network. Of course, the communication link is through a cable and one way but it is a network known as Metropolitan Area Network. Another common communication system is pager system; it is a network like MAN.

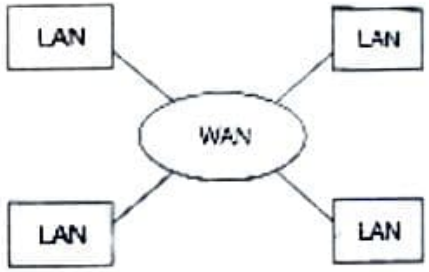


Fig.(5.4) Wide Area Network

(c) Wide Area Network (WAN)

As name itself implies the coverage area is very wide, stations are very high in number hence named as WAN - Wide Area Network. As shown in fig.(5.4) WAN is interconnecting LANs. A very common example is national telephone system, cellular mobile phones, pager etc. WAN is a wide network, which is capable of covering the whole country or state for communications. In WAN, the links may be established by using cables, microwave towers or may be satellite link or it can be combination of two or many links. For example, a telephone system uses all the three media cables, microwave and satellite links for the communications. The railway reservation system, Airline reservation system these are also computer networks treated as WAN.

WAN have following characteristics:

- The area covered is very big.
- The data transfer is slow as compared to LAN.
- The connection media is normally a telephone line to interconnect LANs.

5.3 NETWORK SECURITY

The term network security arises a lot of issues. Some issues are related to software whereas others are related to hardware.

Software security refers to unauthenticated access of programs. Sometimes multiple programs are accessed by different users. When the programs are not stored in memory in a proper way, then the programs can be damaged or data can be lost. Such issues are to be thought in network operating system in such a way that the data should not be lost or programs should be maintained in a proper way. The better technique should be thought and adopted to solve such software security problems.

In the hardware security, the unauthenticated person should not be allowed to access any network resources like floppy drive, hard disk, CD drive etc. The operating system should adopt proper policy regarding such problems.

5.4 NETWORK CONFIGURATIONS

Network configuration means the method of operating a network by deciding the node authorities. There are two basic configurations generally found in many organization

i) Client-Server configuration
ii) Peer to Peer configuration

i) Client-Server configuration: In this type of network many users use a network but a master PC called **'Server'** controls and provides services to these users. The user is making interactions by requesting server to share resources, to access programs data etc. The server is a high performance PC in the group and has highest operating speed, a large memory and a big hard disk size. Each user behaves as a client with server hence known as Client-Server configuration. E.g. Novell Netware

ii) Peer to Peer configuration: In this type of network there is no such relation like server-client but each PC is having equal rights. Each node or Peer is communicating with other Peer directly there is no server hence known as Peer to Peer configuration. The hardware of each peer is almost standard and equal. E.g.Wndows-95, Windows-NT workstations.

5.5 NETWORK APPLICATIONS

In the network, the work can be done in different ways i.e. collaborative, server based etc. As the scope of network is increased and the entire set of network can be accessed by different access methods, it has become possible to collect the information from any place of the world. This is called as Internet. Internet is a worldwide collection of networks. The Internet has different applications

- E-mail
- Voice mail
- FTP
- WWW
- E-commerce
- Chat
- BBS
- User groups

E-mail

E-mail is an electronic mail. The messages can be sent electronically over a network or Internet. For sending or receiving an email, the user must have an email address, email address are given as:

username@location

- Username: it is the recipient's email name.
- @: It is a character, which is used to separate the email name and location.
- Location: it is a place, the electronic post office, where the recipient's mail is delivered and stored.

Nowadays, email services are provided free of cost by a lot of web sites. For example www.hotmail.com, www.rediff.com, etc. The user needs to register with the company. The company gives the user an email account, which indicates the user name with the server name or company name. Using such email addresses, the user can communicate with anybody. The mails can be sent to any person. Only the requirement is the destination person must have an email address. The user can send a text, voice or picture on an email as an attachment.

Voice mail

Initially only text mails were been sent. If the voice information is to be sent or received, then the user has to send it as an attachment file. But nowadays-new technology has come into picture and it is possible to send and receive the voice data directly through Internet as a voicemail. The user need not have to read the messages; a voice output is given on the computer. Only the requirement is that the computer should have multimedia facility and a voice mail software.

FTP (File Transfer Protocol)

It is a fast application level TCP/IP protocol widely used for transferring both text-based and binary files to and from remote systems, especially over the Internet. Most of the people use FTP program for downloading the software on the Internet.

The World Wide Web (WWW)

In the Internet, different types of computers are connected to each other. These computers may have different operating systems. When the data is to be transferred from one computer to other computer, and the operating system of both the computers are different, then both operating systems should understand the data format, which is to be transferred. The www provides an interactive document and the software to access the data on any computer. The document has pages, which can combine text, picture, form, sound, animation and hyperlinks.

E-commerce

Web technologies are playing very important role in business. The web sites are created to perform the business. Online trading is nowadays very important feature of the Internet. It is also called as E-commerce i.e. Electronic Commerce. The businesses are set up on the web. The traditional concept of setting up the business in a place has become an outdated concept. The web sites are created to set a business. People do the merchandise as well as retailing business on the web. So the geographical boundaries have become faint due to Internet and e-commerce.

BBS (Bulletin Board Service)

It is called as Bulletin Board Service. This is also an Internet application. Anyone can set BBS using special software loaded on the machine. The basic use of BBS is that any one can send his/her information on the BBS. The BBS user can get all the information, which is sent by all the other users. The information send on BBS service get broadcasted. So, any one can get the information. People send their requirements on BBS, someone in the world have the information, that person sends the information to the required person. This way anyone can get the information with less effort and in less amount of time.

Chat

It is an Internet application. People used to talk on telephone from distant places But the cost of talking on a telephone line is very high. Using program the user need to get connected in a chat – session and can communicate to anybody in the world. Here the web site have chat program, the chat program have different chat rooms. The user have to log in the chat room, get access in a particular chat room; find other user connected in a chat room and start chatting with the users connected in that room. Simply typing messages for the users can do the chatting.

As this application is very cheap as compared to talk on a telephone line, it has become more popular. People decide a particular time and log into the Internet a start chatting. The cost of chatting as same as a local telephone call.

User Group

User groups are becoming popular service in the Internet. Certain web sites allow the user groups to be created, and communicate with each other independently in a group. In the user group program the user have to register with him by giving his requirement to the group. The groups are created with some specific aim. So, the information related to that a person who belongs to it distributes particular activity in a group. In BBS the information is sent to everybody whereas in user group, the information is given to specific people. So the non-related information can be avoided by using user groups.

QUESTIONS

1. Select the correct alternatives:
   a) The medium size network is……..
      i)LAN ii) MAN iii)WAN
   b) FTP stands for …….
      i)File Transfer Program ii) File Transfer Protocol iii) File Transfer Peer
   c) Interconnecting a set of computers with each other using transmission media is called as ……..
      i)Network ii) Intra-network iii)Workstation
   d) In ……. Configuration server is absent
      i) Client-server ii)peer to peer iii) none of these
   e) WWW stands for …..
      i) World Wide Web ii) World Wise Web iii) World Web Web

   Ans: a) MAN b) File Transfer Protocol
   c) Network d) peer to peer e) World Wide Web

2. Define a network and explain in brief.
3. What are the applications of networking?
4. What is the difference between LAN and WAN?
5. What is the basic difference between Client-Server configuration and Peer to Peer configuration?
5. List various network installations and explain.
6. List different applications of Networking and explain any two.
7. Write a note on:
   a) Chat
   b) E-mail
   c) E-commerce

❑❑❑

SPECIMEN QUESTION PAPER (PAPER-I)

[According to HSC Board Pattern]

Marks: 50 Time:3Hrs.

Q.1A) Select the correct alternatives and rewrite the sentences. 4

a) The Octal number system has Radix of _____
i) 7 ii) 8 iii)10

b) _____ is a graphical representation of program flow.
i) Algorithm ii) psedocode iii)Flowchart

c) Conditional operator is equivalent to _____ control structure.
i)for, do ... while ii) if ...else) iii)none of these

d) In VB for 3D look appearance is _____
i) 1 ii) 2 iii)3

B) Answer any (TWO) of the following. 6

i) Explain 2's complement method of subtraction with suitable example.
ii) What is last step in Program Analysis? Explain its need along with the types.
iii) Explain different bitwise operators in C++.

Q.2A) Answer any (TWO) of the following. 6

i) Convert the following a) $(2AC)_{16} = (\ )_{10}$ b) $(574.321)_8 = (\ )_2$
ii) Write algorithm for finding factorial of a number. The value of number should be accepted from user.
iii) Explain the difference between prefix and postfix operator.

B) Answer any (ONE) of the following. 4

i) Draw flowchart for calculation of fibonacci series
ii) Given the following declarations:
int a = 0, b = 1, c = -1;
float i = 2.5, j = 0.0
Evaluate the following and justify your answers:
i) a > b, ii) b / c < i, iii) a <= b >= c, iv) a <= 1 = = b, v) ++ a = = b ! = j *2

Q.3A) Answer any (TWO) of the following. 6

i) State different types of code systems and explain any one.
ii) What is the basic difference between Structural and Modular programming.
iii) What is function overloading? State its advantages and disadvantages.

B) Answer any (ONE) of the following. 4

i) Explain different control structures in C++ with syntax.
ii) Find X if i) $(2AC)_{16} = (X)_2$ ii) $(1110.1101)_2 = (X)_{16}$

Q.4) Answer any (TWO) of the following. 6

i) Write special features of Visual Basic.
ii) What is the difference between LAN and WAN?
iii) Explain how to create a new project in VB.

B) Answer any (ONE) of the following. 4

i) What is pointer? Write its advantages.
ii) What is array? Explain multidimensional array.

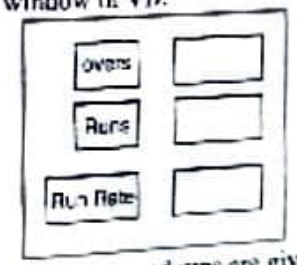
Q.5) Answer any (TWO) of the following. 10

i) Write a program in C++ to find the sum of series 1+3+5+...+n the value of n should be accepted from user.
ii) Develop a student's result system in VB which will accept roll no., name, and marks in all subjects and calculate total and percentage and determine class of student.
ii) Write a program in C++ to find area of circle by using return type function.

OR

Q.5) Answer any (TWO) of the following. 10

i) Design a scoreboard window in VB:



It should calculate run rate if overs and runs are given and if overs > 50, it should display proper message.
ii)Write a program in C++ to display Odd Numbers between 1 – 100 using For Loop.
iii) Write a program in C++ to display

```
*
**
***
****
```

□□□

COMPUTER HARDWARE

(PAPER-II)

1. ELECTRONIC COMPONENTS

INTRODUCTION

The application of electronic science is widely growing due to large variety of electronic components. Most of the success in electronic field over electricity is a due to various characteristics of components, like nonlinear performance, integration, cost and size of the electronic components. Without studying theoretically and practically their basic construction and function, it is difficult to understand working of circuits. In this chapter most of the common electronic components has been discussed up to a certain extent, so that students can understand basic action of the components, their ratings, colour code systems and desired practical knowledge of component testing.

1.1 CLASSIFICATION OF COMPONENTS

If you observe carefully any electronic circuit, you can be a little surprised: there are only five varieties of the electronic components. Actually the circuit may appear complicated because of number of different components, but each type belongs to any of these five types. Electronic components which are now available in the market are classified into two groups (A) Passive Components (B) Active Components.

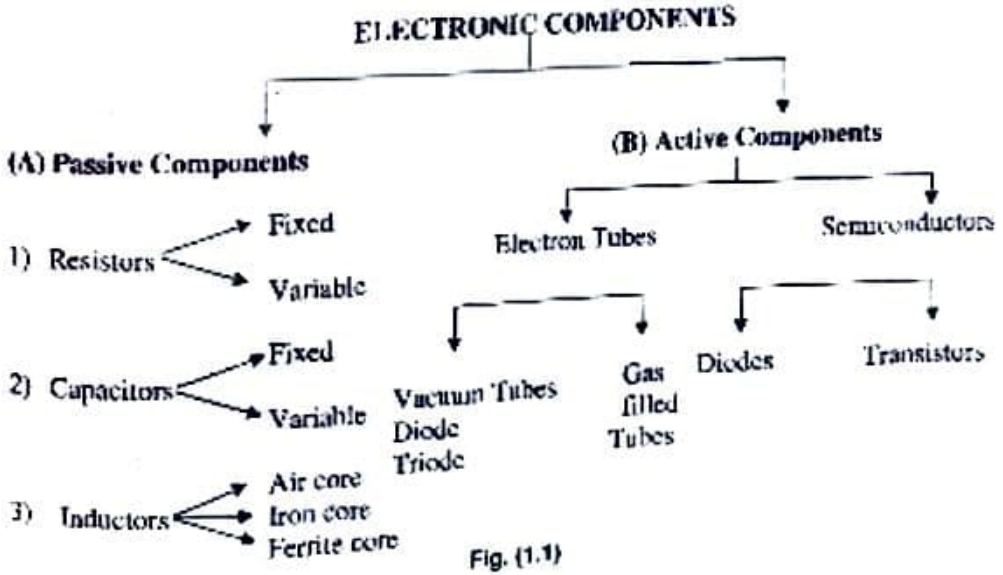


Fig. (1.1)

The above chart shows classification of components in brief. Detail classification can be made in different ways.

(a) **Passive Components**: Since this category belongs to inactive functions of the component, these components do not have capacity to amplify the voltage or to rectify supply. Therefore, they are called as *"passive components"*. However, without these components assembly of electronic circuit is not possible. Resistors, Capacitors and Inductors are called as passive components.

(b) **Active Components**: Since these components are capable of performing active functions like amplification, rectification and switching, they are called as *"active components"*. Hundreds of active components are now manufactured they can be classified in to two broad groups (i) Electron tubes like Diode, triode and (ii) semiconductor devices like diode, LED, Transistor etc. Electron tubes are now rarely used because they are outdated.

1.1 RESISTORS

In electronic circuits Copper, Aluminum materials are known as conductors, having negligible resistance. Other materials like paper, ceramic, are known as insulators, having very high resistance. Other materials like carbon with certain value of resistance is used from very low to high value between conductor and an insulator. These materials are called as resistors. The value of resistance is measured in Ohm (Ω). Resistance is the opposition to the flow of electrons or simply opposition to the electric current. Resistance is required in electronic circuits to **limit the current, to drop the voltage** and **to divide the voltage**. In combination with capacitor, it is used as filter or it can be used to achieve the time delay.

TYPES OF RESISTORS

Resistors may be classified as (a) Fixed Resistors and (b) Variable Resistors. The resistors are manufactured by either of three basic materials-carbon composition, wire wound and metallized. These materials differ in their resistance material. According to their construction, resistors may be fixed or variable. Fixed resistors, whose resistance value is constant while the resistance value of variable resistor is adjustable between the specified range. The symbol of fixed and variable resistors is shown in fig (1.2)

Fig. (1.2) Symbols of Resistor

(a) FIXED RESISTORS

In this category, resistance has only one value and most of the structure of this type is cylindrical in shape that has two wire leads or terminals. Fixed resistors can be classified into three types according to the material used

(1) Carbon types → (a) Carbon composition type, (b) Carbon film type.
(2) Wire wound type
(3) Metal film type

Let us see the construction and special features of each in brief.

(1) Carbon Composition Resistors

These resistors are widely used in electronic circuits. Carbon composition resistor's value is available from 1Ω to about 20 MΩ and typical power ratings are 1/8 W to 2W. Speciality of these resistors is low cost and small size. (Most of the resistors available in your laboratory are carbon composition type). These resistors are manufactured by mixing granules of carbon with binding material. This cylinder is molded in a cylindrical form as shown. Fig. (1.3) shows the construction and different wattages of resistors. Tinned copper wire leads are inserted in the two ends and structure is sealed with non-conducting coating. The cost of resistor depends on its wattage and not on the value of resistor.

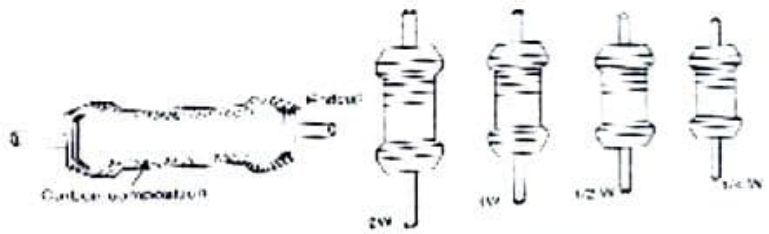


Fig. (1.3) Carbon Composition Resistor

Colour Coding of Carbon Resistor

Another specialty of carbon resistor is its colour code system, used to indicate its value and tolerance, where tolerance specifies the maximum deviation of the actual value on both sides. Hence it is expressed in terms of % , e.g. if a resistors of 100 Ω value and its tolerance is ± 5 % , this means the resistor value may be in the range of 95 Ω to 105 Ω . If it is ±10% then its value is in between maximum 110 Ω to minimum 90 Ω and so on. If user wants very close value to 100 Ω then he will select ± 5% tolerance. Resistor of ± 5% tolerance indicates that its value may not be exactly 100 Ω it lies in between ± 5% of the actual value. Tolerance is specifying % error in actual value.

Method: In colour code method, the first two rings indicate the first two digits and the third ring indicates the multiplier. Fourth ring is used to indicate its tolerance.

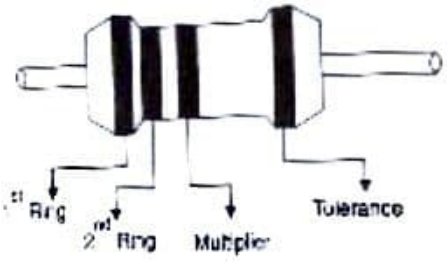


Fig. (1.4) Colour Code Method

Colour Code Chart

Colour	Value of the 1st and 2nd ring	Value of 3rd ring (Multiplier)	Tolerance
Black	0	X 10^0	
Brown	1	X 10^1	±1%
Red	2	X 10^2	
Orange	3	X 10^3	
Yellow	4	X 10^4	
Green	5	X 10^5	
Blue	6	X 10^6	
Violet	7	X 10^7	
Gray	8	X 10^8	
White or No Colour	9	X 10^9	±20%
Gold	-	X 10^{-1}	±5%
Silver	-	X 10^{-2}	±10%

Table (1.1)

Examples: Find the value of carbon resistors by using colour code method.

i) Red, Violet, Red Gold
2 7 $X10^2$ ±5%
= 2 7 X100 =2700Ω =2.7KΩ Or 2K7

ii) Brown, Black, Orange Silver
1 0 $X10^3$ ±10%
= 10 X1000=10,000Ω =10KΩ

iii) Red, Red, Gold Gold
2 2 $X10^{-1}$ ±5%
=2.2÷10 =2.2Ω

iv) Green, Black, Yellow Silver
5 0 $X10^4$ ±10%
=50X10000=500,000Ω=500KΩ=0.5MΩ

Speciality – Low cost, colour code system is used.
Range - 1Ω to 20 MΩ, Tolerance - ± 1% to ± 20%, Wattage- 1/8 W to 2W

(2) Wire Wound Resistors

Wire wound resistor is used where high current control is required as well as where stable and accurate value of resistor is required. A wire wound resistor is manufactured by wrapping a length of special resistive wire like nichrome or mangenin around an insulating core of ceramic. The end of wire is attached to metal contacts, which are inserted in the core. After wire is wound, the whole structure is coated with enamel containing powdered glass and then heated to produce coating, this is known as vitreous enamel coating. It is required to protect the resistor and to dissipate the heat. Typical values are available in fraction of ohms to 100 KΩ. Power rating is from 5 W to 200 W.

The drawback of this resistor is its big size and higher values are not possible due to its size.

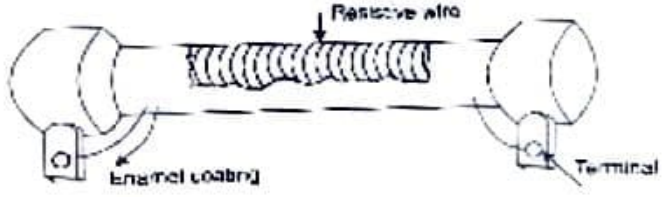


Fig.(1.5) Wire wound Resistor

Speciality : Big in size, More power handling capacity. Colour Code System is not used
Range : Fraction of Ω to 100 KΩ and power rating 5 W to 200 W.

Comparison:

Carbon Composition Resistor	Wire Wound Resistor
1. Carbon granules are used.	A resistive wire like nichrome is used.
2. Tolerance is ±2% to 20%	More precision type because tolerance is very low.
3. Wide range of resistance.	Available in low values only.
4. Low power ratings.	High power ratings.
5. Color code system is used.	No color code system.
6. Small size & low cost.	Big size & high cost.

(B) VARIABLE RESISTORS

Variable resistors are commonly called as rheostats. In low current circuit like in electronic circuit, small variable resistors are called as "*potentiometer or pots*". This potentiometer name probably originated from the work "*Potential divider*". Variable resistors are normally used in radio or audio equipment for volume control, contrast and brightness control in TV sets. According to material used these resistors are classified into two types
(1) Carbon type and (2) Wire wound type

(1) Carbon Type Potentiometers

Basically the construction of all potentiometers is same. In this type a movable metal pointer known as a 'wiper' moves on a circular carbon resistive track as shown in fig (1.6). This wiper is attached to a shaft it may be metal or a plastic shaft.

Fig.(1.6) Idea of Potentiometer

Resistive carbon material is deposited on a Bakelite piece. Variable resistors always have three terminals as shown, terminal 1 and 3 are fixed terminals and 2 as variable terminal. When terminal 1 and 3 are connected in the circuit you will get fixed and maximum resistance. When 1 and 2 or 2 and 3 terminals are connected you will get variation in resistance when you rotate the shaft.

If you connect terminal 1 and 2 at extreme left side wiper will be at terminal 1, hence you will get zero resistance and when you rotate towards the right side its resistance increases towards right side. When shaft is rotated fully, you will get maximum resistance.

These variable resistors are available in two types,

(1) Linear Type (2) Log Type.

In linear potentiometer resistance varies with linear rate. If you rotate the shaft approx for 10^0 rotation it gives 100 Ω at 20^0 → 20 Ω and so on. But in nonlinear potentiometer (log type) resistance varies in log scale, at 10^0 → 10 Ω , 20^0 → 100 Ω 30^0 → 10,000Ω .

When the maximum range of resistance is required 'Log type' pot is preferred. Linear type pots are marked with "LIN 10K and log type with "LOG 10K ". When often variation of resistance is required these carbon potentiometers are used like volume control but when rare adjustment is required small potentiometer called as *"presets"* are used. The variation of resistance is done with a screwdriver. The internal construction is similar to that of potentiometer.

Speciality : Low Cost, Used in Radio, TV as volume control, brightness control etc.
Range : 1 KΩ to 5 MΩ, Power rating 1/2W to 2W

(2) Wire Wound Potentiometers

Basic principle of construction of wire wound variable resistor is simple: the most common example of this type is rheostat. In this type wire is wound on an open ring of ceramic, which is covered with vitreous enamel.

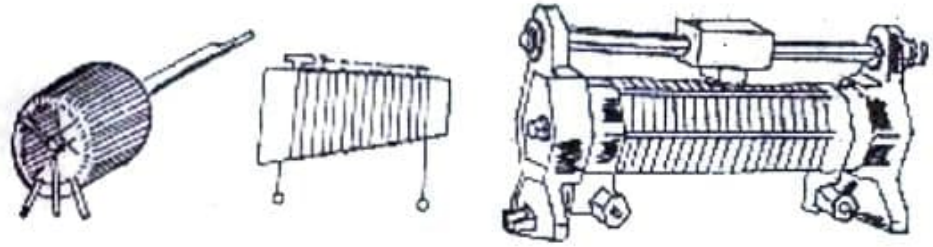


Fig. (1.8) Wire wound Potentiometers

The specialty of such type is high power handling capacity and very small and accurate change of resistance. Figure shows the construction of wire wound pot and rheostat. In wire wound variable resistor non-linear resistor is also available (tapered type). Non linearity in the resistor is achieved by using tapered strip on which resistive wire is wound. Fig. (1.8) shows linear and non-linear potentiometers. By sliding the contact, the variation of resistance is achieved.

Speciality : High Cost, More power handling capacity. Big size.
Typical Range: 100Ω to 1 MΩ, Power rating 2 W to 100 W.

(C) OTHER RESISTORS

Because of large development in semiconductor technology in addition to diode and transistors; special type of resistors are manufactured which may be called as active resistors.

(1) Thermistor

A Thermistor is a semiconductor device whose resistance varies with temperature but change in resistance is non-linear. Conductors show very small change in resistance but thermistor can increase or decrease its resistance by more value. Hence thermistor can be NTC or PTC type. When resistance decreases with increase in temperature then it is called as "Negative Temperature Coefficient" of resistance (NTC) and if it increases then it is called as "Positive Temperature Coefficient" of resistance (PTC). Thermistors are used in many applications like measurement of temperature, temperature control, temperature compensation etc.

(2) Light Dependent Resistor (LDR)

It is a photosensitive device whose resistance decreases with increase in light intensity. The symbol is shown in fig. (1.9) Cadmium sulfide material is used for this purpose. Its dark resistance is about 100KΩ and it decreases with increase in light. LDR is widely used in many applications like automatic street light control, burglar alarm etc.

(3) Varistor (VDR)

It is also a non-linear resistor in which resistance is made variable to voltage. It is also known as voltage dependent resistor. They are two types, (i) Silicon Carbide VDR, (ii) Metal Oxide VDR. Varistors are used in industrial applications and TV circuits for surge suppression.

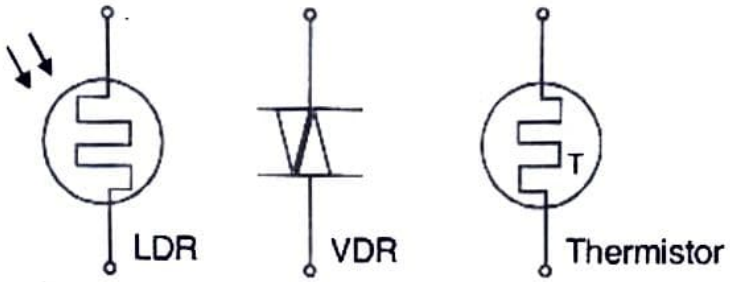


Fig. (1.9) Active Resistors

1.2 CAPACITORS

Capacitor is second important passive component, commonly observed in electrical and electronic circuits. Many times it is called as condenser. Basically it is used to store electrical energy (by charging) and to deliver this stored energy (by discharging). This component has specific capacity of storing charge across it, hence named as capacitors. They are available in fixed and variable types; their capacity is measured in **'farad'**. They are available in typical range of pico farads to microfarads.
A capacitor consists of two parallel conducting surfaces separated by an insulator called as dielectric material. Different symbols of capacitor are shown in the fig. (1.10).

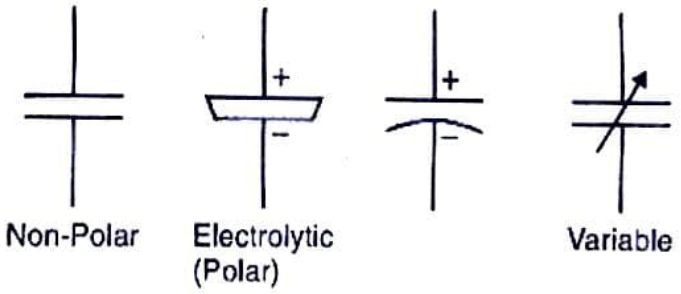
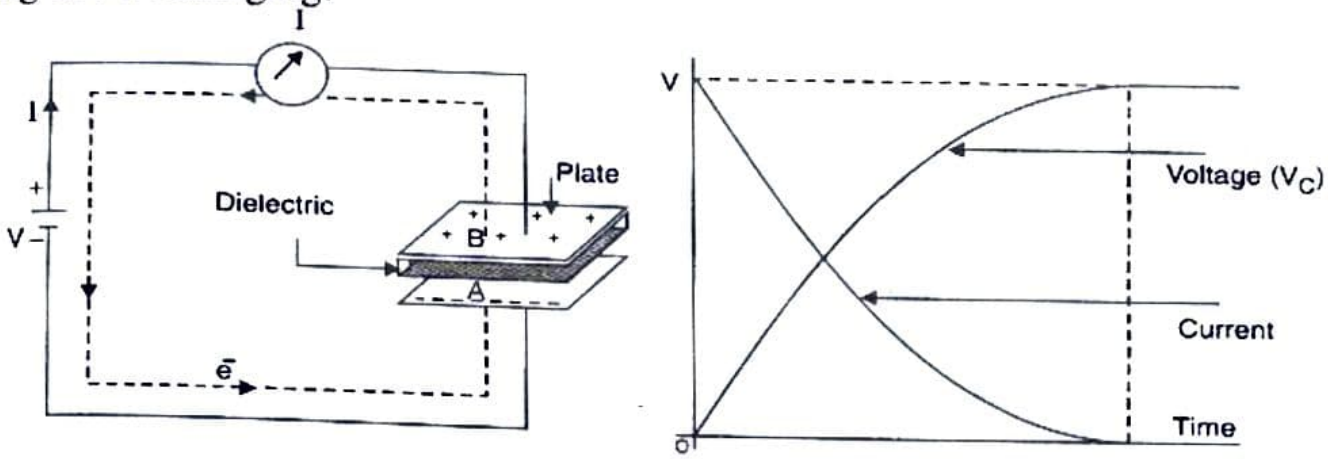


Fig.(1.10) Symbols of Capacitor

A capacitor is formed by two parallel conducting plates (may be spherical in structure) separated by an insulating material. Insulators are like mica, paper, ceramic called as *"dielectric"*. In electronic circuits capacitor plays an important role like resistor, it has got many applications such as in photo camera for flash, in power supplies used as

a filter, in amplifiers as a coupling or a bypass capacitor and so on. Let us see its action of charging and discharging.



(a) **Fig. (1.11) Capacitor Charging** (b)

As shown in the fig.(1.11a) when battery is connected across the capacitor, current flows through the circuit, current meter is connected for the indication. Before connecting the battery the number of electrons on plate A and B are equal but when battery is connected across it, electrons are accumulated on plate A because they cannot flow through dielectric. An electrostatic force is developed by the plate A gives a pressure to plate B. It passes the same number of electrons towards the battery. This is known as electrostatic induction. Thus we get negative charge on the plate A and positive on the plate B that produces a potential difference, electrons continue to flow till voltage across capacitor becomes the battery voltage. Fig.(1.11b) shows how capacitor voltage increases towards battery voltage and it shows current through it.

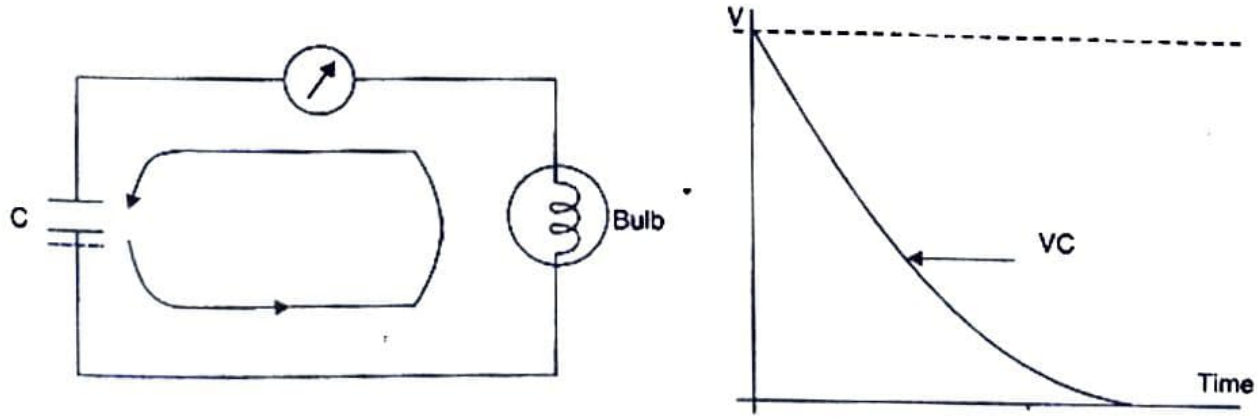


Fig. (1.12) Discharging of capacitor

Now voltage across capacitor is `V' it can be discharged through a resistance as it is shown in the fig (1.12) through the lamp. The lamp glows for a short time and then it becomes OFF because voltage across capacitor decreases from V to zero as shown in the graph.

RC Time Constant

This is an interesting term, which is related with timer circuits. As discussed earlier capacitor charges to applied voltage but how much time it takes to charge up to battery voltage? On the other hand when capacitor is discharged what is the time required to get total discharge? There is a relation, which is expressed by the term "*Time constant*". It is defined as

"The time constant is the time in which capacitor gets charged up to 63.28% of applied voltage or it is the time in which capacitor gets discharged up to 36.72% of its initial voltage."

Where T is calculated by formula

$$T = R \times C$$

It means the time constant depends upon the value of resistance and capacitance.

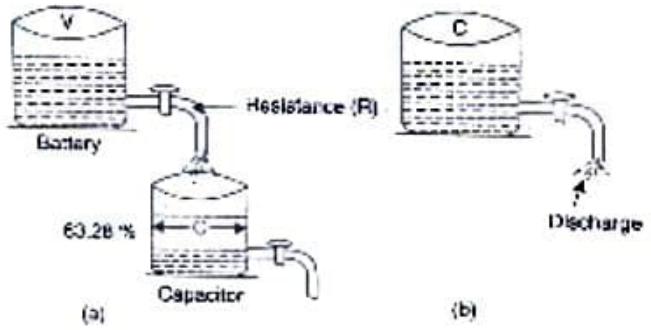


Fig. (1.13) Analogy for charging and discharging

The fig. (1.13) illustrates the idea of charging and discharging with analogy of water. Capacitor charging is a process like filling water in a tank. The time required to fill the tank depends upon the flow of water i.e. resistance (R) and capacity of the tank i.e. capacitance (C). Similarly, as shown in the figure (b) the discharging process a similar to discharging water from the tank, it also depends upon the flow of water and capacity of the tank.

For example, if a capacitor of 5 μf is charged through a resistance of 2MΩ and through 10V battery then 1 time constant is

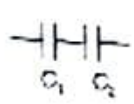
$$T = R \times C = (2 \times 10^{6}) \times (5 \times 10^{-6}) = 10 \text{ sec.}$$

In 10 sec time capacitor will get charged up to 63.28% of the battery voltage that is up to 6.328 V and in 5-time constant $5 \times 10 = 50$ sec it will charge up to 10 volts

Remember: In 5 time constant capacitor gets charged to applied voltage or in 5 time constant it will get complete discharged.

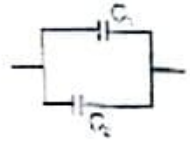
Formulae: a) Capacitors in Series

$$\frac{1}{C_S} = \frac{1}{C_1} + \frac{1}{C_2} + \frac{1}{C_3} + \ldots\ldots$$



b) Capacitors in Parallel

$$C_P = C_1 + C_2 + C_3 + \ldots.$$



Example:

1) Two capacitors 20μf and 5μf are connected in series and then connected in parallel the effective capacitance in each case.

Solution:

In series

$$\frac{1}{C_S} = \frac{1}{C_1} + \frac{1}{C_2} = \frac{C_1 C_2}{C_1 - C_2} = \frac{20 \times 5}{25} = 4\mu f$$

In parallel

$C_P = C_1 - C_2 = 20 + 5 = 25\mu f$

2) A capacitor of 0.2μf is connected in series with a resistance of 15MΩ. Find the time constant and the time required to charge up to battery voltage if battery voltage is 10V. What is the voltage across capacitor at 1 time constant.

Solution:

$T = R \times C = [15 \times 10^{6}] \times [0.2 \times 10^{-6}]$

$= 3$ sec.

After 3 sec. The voltage across capacitor is 6.3 volts. It takes $3 \times 5 = 15$ seconds to charge up to 10 volts

TYPES OF CAPACITORS

Capacitors can be classified according to the dielectric material used. Polar, non-polar and Fixed/variable. They are classified as

(A) Fixed Capacitors

(B) Variable capacitors.

A) FIXED CAPACITORS

Fixed capacitors can be classified according to dielectric material and polar / nonpolar such as,

(i)	Paper capacitor	Non polar capacitor
(ii)	Mica capacitor	Non polar capacitor
(iii)	Polyester capacitor	Non polar capacitor
(iv)	Ceramic capacitor	Non polar capacitor
(v)	Aluminium electrolytic capacitor	Polar capacitor

Let us see the special features, typical range and construction of each capacitor in brief.

(1) Paper Capacitors

Basically the construction for all fixed non-polar capacitors is same. Two conducting surfaces are separated by dielectric material. Paper capacitor is manufactured by rolling two thin metal foils (foil is a long strip of metal) separated by a paper roll as shown in fig.(1.14). Paper used for this purpose is Kraft paper impregnated with wax or resin or oil. Where, impregnation is the process in which paper is soaked with dielectric wax or oil.

This is a non-polar capacitor because its polarity is not important. It can be connected in either way in the circuit. Polarity is not mentioned on the structure of these capacitors.

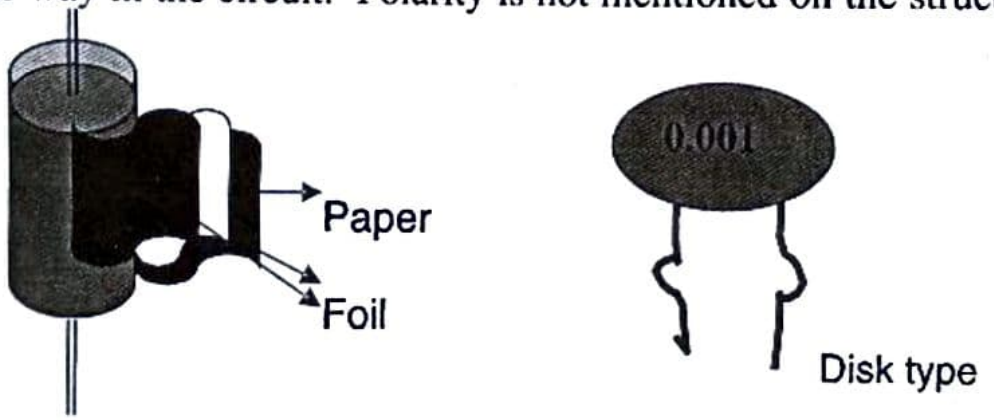


Fig (1.14) Construction of Paper Capacitor

As shown in figure two foils are rolled by placing impregnated paper in between them to form a cylindrical structure. Two wire leads are attached to the foils as the connecting terminal. Metal plates used here, are in the form of foils to increase plate area and to minimize the size. These capacitors are available in molded and tabular type as shown in fig.(1.14)

Speciality : Medium range of capacitance, small size and non-polar.

Typical range : 500pF to 1μF and typically voltage rating from 100 V to 1000V.

(2) Mica Capacitors

The construction is similar to that of paper capacitor only the dielectric material used is mica foil or mica sheet as shown.

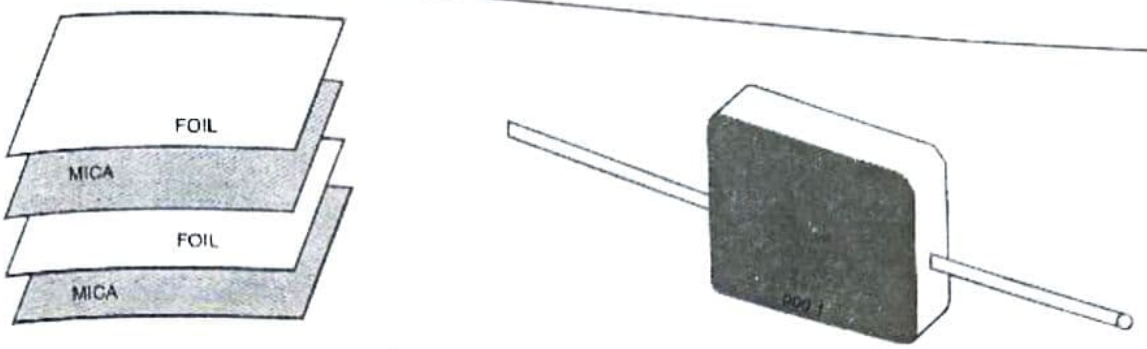


Fig. (1.15) Mica Capacitor

Mica capacitors are square or rectangular in shape with tined wire leads. The entire structure is molded in Bakelite sheet.

Speciality : High temperature range – 55°C to ± 150°C Non-polar.
Typical range : 5 pF to 10,000pF Voltage rating about 500V.

(3) Ceramic Capacitor

These capacitors are commonly observed in many circuits like amplifier, oscillators etc. as coupling capacitor. They are available in many different shapes like disc type, rectangular type, tabular and button type as shown. It is also a non-polar capacitor.

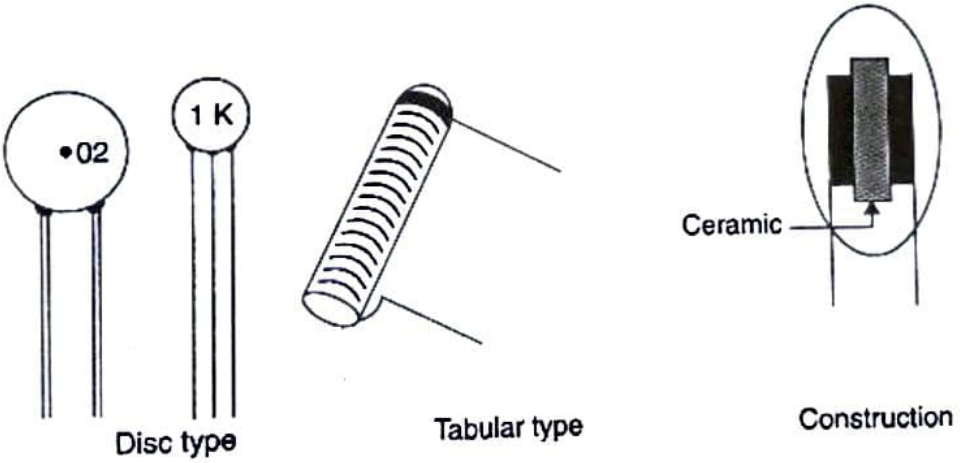


Fig (1.16) Ceramic Capacitors

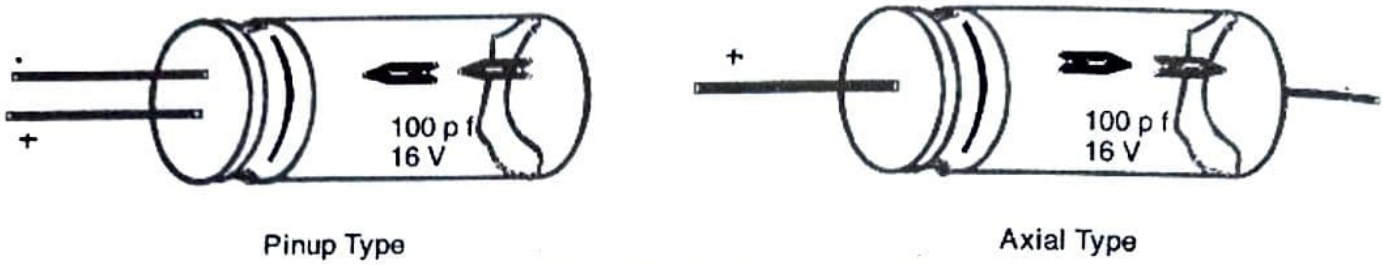
The construction of ceramic capacitor is different than paper and mica capacitor. It is constructed by depositing silver plates on each side of a very thin disc of ceramic material. During the process tinned wire leads are attached to each plate.

Speciality : Most economical in price 1 Rs. per piece.
Typical range : 3pF to 2 μF and voltage range from 3V to 6000 Volts non-polar.

(4) Aluminium Electrolytic Capacitor

This is a polar capacitor on which polarities are marked as positive and negative. Polarity is important because reverse connection may damage the capacitor. Another

main difference between non-polar and polar is; these capacitors are manufactured in higher values. Higher values are possible because electrolyte is used to provide more capacitance in smallest space.



Pinup Type Axial Type

Fig (1.17) Electrolytic Capacitor

Electrolytic capacitor consists of Aluminium foil electrode as an aluminum oxide film covering one side and is in contact with an electron conducting fluid known as an electrolyte. The Aluminium oxide forms a dielectric and the electrolyte form another plate of capacitor. A layer of plain Aluminium foil is also wound into the capacitor to provide electrical contact between one of the terminals and electrolyte.

Normally electrolytic capacitors are not used in AC circuits mostly used in DC circuits like filter in rectifier.

Speciality : Higher values, High cost, Polar capacitors.
Typical Range : 1 μ f to 3000 μ f and voltage rating 1V to 500V.

(B) VARIABLE CAPACITORS

There is a need of variable capacitor in various circuits like tuning circuit of radio, oscillator circuit to vary the frequency etc. Air is the most common dielectric used in variable capacitor.

(1) Air Gang Capacitor

Air gang condenser is a common example of variable capacitor. Other variable capacitors are similar to variable condenser but they are adjustable known as trimmers and padders. Gang condenser consists of two set of parallel plates. One of the sets is fixed called as stator and another set is called as rotor. This rotor set of parallel plate is fitted on a movable shaft. When the shaft is rotated the area covered by each plate of rotor and stator is varied and thus total capacitance is varied. In valve type old radio circuit two gang capacitors are used while in transistor receiver miniature type is used. Fig. (1.18) shows simple construction and other variable air capacitors.

Speciality : Used for tuning the radio, large size, high cost but high operating voltage.
Typical Range : 1pF to 500 pF. Maximum operating voltage 1 KV.

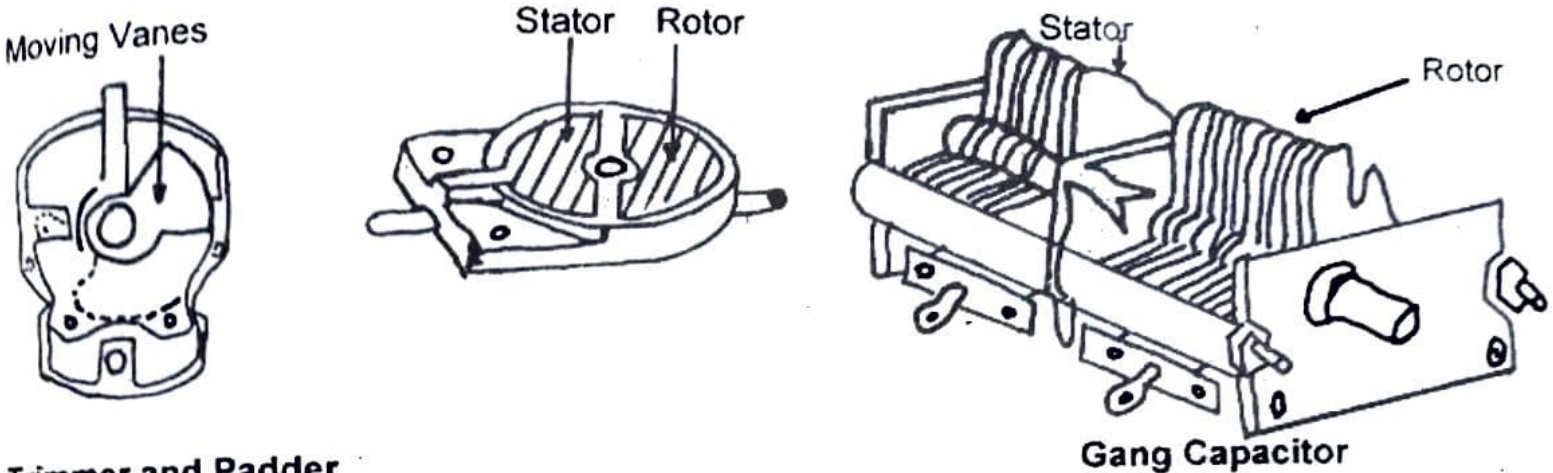


Fig (1.18) Variable Air Capacitors

1.3 INDUCTORS

Inductor is a third important passive component widely used in AC circuits than DC. Inductor is a simple coil of copper wound on a core material.

Inductance is the ability of a conductor to produce an induced voltage, when current through it varies. In other words inductor opposes change in current.

The unit of inductance is **"Henry."** It is defined as the inductance that will develop a voltage of one volt across it when current changes at the rate of one Ampere per second or when a voltage of one volt is induced for the change of current rate 1 Amp / sec then the inductance of coil is said to be 1 Henry.

Inductance

It is the property of conductor to produce induced voltage, when current through it varies. Induced voltage is the result of change of flux across a conductor because when AC current flows through it magnetic flux varies its strength and the direction, which is equivalent to motion of the flux. When DC current flows through the conductor voltage induction is not possible because DC current is steady and it flows only in one direction. Inductor behaves like a conducting wire on DC.

Types Of Inductors

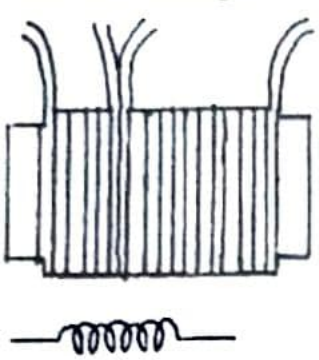
Inductors can be classified into two groups as, (i) Fixed inductors (ii) Variable inductors. But they are classified according to the material used for the core.

(i) Air core inductors (ii)Iron core inductors (iii) Ferrite core inductors

(i) **Air core Inductor:** In this type a length of conductor, copper wire is wound on a hollow tube of these material. Here air acts as a core, hence the name air-core inductor.

(ii) **Iron Core Inductor:** Here iron is placed as former of the coil so known as iron core inductors. Iron is a magnetic material, so when AC current flows through a coil the

lines of flux are produced. These lines of flux pass through the iron core without any external leakage. The iron core inductor is commonly called as a choke.



Air Core Inductor

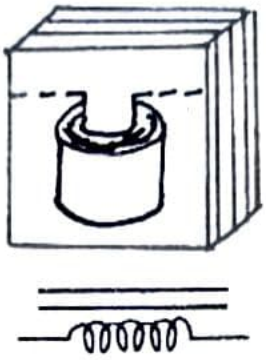
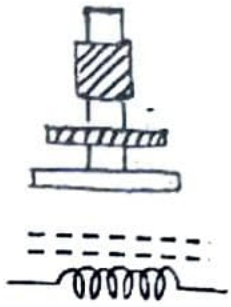


Fig (1.19) Iron Core Inductor



Ferrite Core Inductor

iii) Ferrite Core Inductor: Ferrite is an artificially made non-metallic material. When a Ferro-magnetic material is placed in former of the coil the Ferrite core exhibits a minimum loss of lines of flux. .

1.4 TRANSFORMERS

A transformer consists of two inductors. One of these inductors is meant for supplying alternating current from an external source and is referred as a primary winding. In transformer, the change in current in one coil can induce an e.m.f. in the second coil. The coil in which the voltage is induced is known as the secondary winding. Hence, it is a device, which transfers alternating currents of high voltage to low voltage and vice-versa.

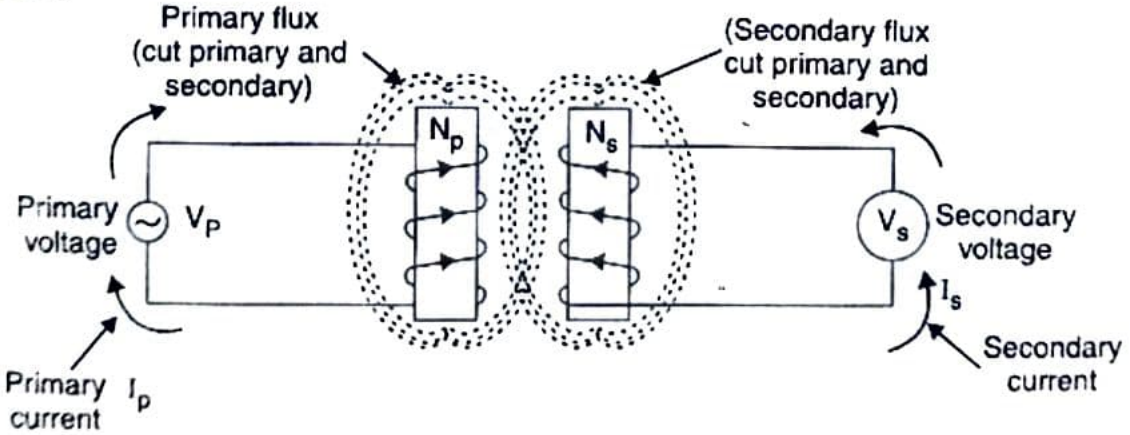


Fig. (1.20) Transformer construction

As shown in the fig (1.20) transformer consists of two windings (coils). The winding which is normally connected to the mains is called as *`Primary winding'* and other winding across which output is obtained is called as *`Secondary Winding'* Both windings are wound on a core, which is laminated type.

Transformer is an important device in which voltage is transformed from one voltage level to other required level. For example, 230 Volt to 6V, 230 V to 9 V etc. This type of transformer is called as ***`step down'***(where Np>Ns) transformer. Similarly 230 V to 1

KV, 230 V to 440 V this type of transformer is called as `***step-up***'(where Np<Ns) transformer. Transformer action is based on Faraday's law of electromagnetic mutual induction. Although the primary and secondary windings are not connected to each other power in primary is coupled into the secondary by the magnetic linking.

Working (Principle)

When an AC voltage is applied across its primary winding, AC current flows through it, therefore changing magnetic flux is produced. Secondary winding is kept close to the primary, which will cut this changing flux. According to Faraday's law, voltage is induced in the secondary winding. The amount of induced voltage depends on the number of primary and secondary turns and that can be calculated by a ratio called as turns ratio.

$$\frac{N_P}{N_S} = \frac{V_P}{V_S} \quad \text{..... Turns ratio}$$

Where N_p = Number of turns in primary Ns = Number of turns in secondary

V_p = Primary Voltage Vs = Secondary voltage.

Example 1 : A power transformer has 100 turns of primary and 5 turns of secondary, calculate i) secondary voltage if primary voltage is 120 volts and
ii) frequency of secondary if primary voltage is 50 Hz.

Solution :

$$\text{Turns ratio} \quad \frac{N_P}{N_S} = \frac{V_P}{V_S}$$

$$\therefore V_S = \frac{V_P \times N_S}{N_P} = \frac{120 \times 5}{100}$$

Secondary Voltage = 6V

Frequency remains unchanged therefore it remains 50Hz.

Example 2 : A transformer has 120 turns of primary and 30 turns of secondary, calculate i) secondary voltage if primary voltage is 220 volts and
ii) load current if load is 1KΩ.

Solution :

$$\text{Turns ratio} \quad \frac{N_P}{N_S} = \frac{V_P}{V_S}$$

$$\therefore V_S = \frac{V_P \times N_S}{N_P} = \frac{220 \times 30}{100}$$

Secondary Voltage = 66V

Load Current = Vs/R =66/1KΩ=66mA

QUESTIONS

1) **Fill in the blanks**

a) For a PTC thermistor the resistance value .....with decrease in temperature.
b) .....resistor is used in high current circuits.
c) LDR Stands for .....
d) Electrolytic capacitor is a ..... capacitor (polar / non polar)
e) In Air gang capacitor ..... is used as dielectric.
f) Transformer works only on .....
g) Inductance is measured in .....
h) Rheostat is a ..... resistor (fixed/variable)
i) If two capacitors each of 50 μf are connected in series its resultant capacitance becomes .....
j) The colour code of 33 ohm resistor is.....

Ans: a) decreases. b) wire wound, c) Light Dependent Resistor, d) Polar, e) air, f) AC, g) Henry h) variable i) 25 μf j)Orange, Orange black

2.State TRUE or FALSE

a) Paper capacitor is a polar capacitor.
b) VDR stands for Variable dynamic Resistor.
c) Diode is an active component.
d) MFR stands for Metal Form Resistor.
e) Electrolytic capacitor is a polar capacitor.
f) Gang capacitor is a fixed type of capacitor.
g) Wire wound resistor is prepared with nichrome wire.
h) Cadmium sulphide can be used to produce LDR.

Answers: a) false b) false c) true d) false e) true f) false g) True h)True

3. Compare between wire wound and carbon composition resistors.
4. Enlist the types of capacitors and explain any one of them in detail.
5. Explain charging and discharging of a capacitor.
6. What do you mean by MFR ? Explain.
7. Explain the construction of paper capacitor.
10. What is thermistor ? Explain the working of N.T.C. and P.T.C. thermistors
11. Explain the construction of electrolytic capacitor.
12. Define 1 time constant and write its equation.

13. How does Mica capacitor form explain with diagram?
15. Write different types of inductors and explain any one.
16. Explain the working of a transformer with its construction.

PROBLEMS

1. Give the colour coding for the following resistance values

1) 1 M Ω 2) 3.3 Ω 3)8200 Ω 4)150 Ω 5) 68 Ω 6) 2 K Ω

Ans: 1) Brown, Black, Green 2) Orange, Orange,Gold
3) Gray,Red Red 4)Brown, Green, Brown
5) Blue, Gray,Black 6)Red,Black,Red

2. Give the resistance and tolerance for the following resistors with colour strips.
i)Yellow, violet, yellow and Silver.
ii)Red, Red, Green and No colour.
iii)Orange Orange , Black and Gold. iv)White, Brown Brown and gold.

Ans: i) 470KΩ ± 10% ii) 2200K or 2.2M Ω ± 20% iii) 33Ω ± 5% iv) 910Ω ±5%

3. Find RC time constant if R = 100 KΩ and C = 100 μ f (**Ans:** 10 sec.)

4. Find the total capacitance if two capacitors of 20 μf and 5 μf are connected in series and if they are connected in parallel. (**Ans:** 4 μf ,25μf,)

5. Which capacitor is required in series with 12 Farad to get an equivalent capacitance of 4 Farad. (**Ans:** 6 Farad)

6. Find the equivalent capacitance if three capacitors each of 12 μf are connected in series and if they are connected in parallel. (**Ans:** C_S = 4μf, C_P = 36μf)

7. Two condensers of 20μF and 5μF are connected in series and a third condenser of 6μF is connected in parallel with this combination find the equivalent capacitance (**Ans:** 10 μFarad)

8. A transformer has turns ratio of 10:1. Find the secondary voltage if the primary is connected to 220V supply. (**Ans:** 22V)

9. A transformer has turns ratio of 5:1 and if secondary voltage is 40V/ 50Hz. find the primary voltage and the frequency of supply. (**Ans:** V_P = 200V, f = 50Hz.)

10. A transformer has 30 turns of primary and 120 turns of secondary, calculate its primary voltage if secondary voltage is 66 volts (**Ans:** V_P = 220V)

1.5 SEMICONDUCTOR DEVICES

In this part of this chapter brief theory of semiconductors is explained to understand the construction and working of diode and transistors. Semiconductors are tetravalent atoms known as **intrinsic semiconductors.** Their conductivity lies in between conductors and insulators. Practically they are prepared by adding impurity atoms to form P-type and N-type semiconductors. This process of adding impurity is known as **doping**. The doped semiconductors are known as **extrinsic semiconductors.** In semiconductors current flows due to motion of two types of charges i.e.Electrons and Holes. Let us see the difference between N-type and P-type semiconductors.

Comparison

N-Type Semiconductor	P- Type Semiconductor
1) Doped with pentavlent atoms.	1) Doped with trivalent atoms.
2) Electrons are majority charge carriers.	2) Holes are majority charge carriers
3) Holes are minority carriers	3) Electrons are minority carriers
4) Pentavalent atoms give electrons therefore called as donor atoms	4) Trivalent atoms create holes, therefore called as acceptor atoms.
5) Antimony, Arsenic, Phosphorus impurity is required	5) Aluminum, Boron, Indium, Gallium impurity is required.

1.4 P-N JUNCTION DIODE

Semiconductor diode is similar to a valve diode; P-type semiconductor is joined with N-type semiconductor so it forms a P-N junction diode. P-type material is acting as because it contains majority holes anode and N type as cathode due to majority of electrons. Fig (1.21) shows this structure and symbol of P-N diode.

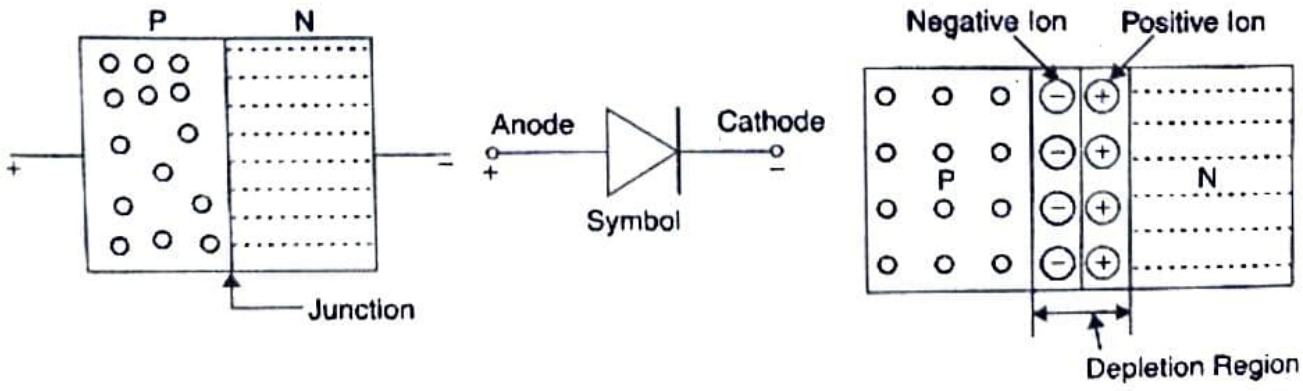


Fig (1.21) P-N Junction diode

Barrier Potential

As soon as junction is formed some of the electrons from N-region near the junction cross the junction and fill some of the holes of P-region as shown in fig (1.5). Since the atoms of N-region loose electrons they become positively charged ions and since the atoms of P-region accept electrons they become negatively charged ions. Thus opposite types of ions are produced (Positive ions on N-side and negative ions on P-side) near the junction as shown in fig (1.21). These ions oppose further flow of electrons from N to P region and flow of holes from P to N region. This opposing force of ions is called as barrier potential and this region is called as **'depletion region'**. This barrier potential is a **0.7V** in case of silicon diode and **0.3V** in case of Germanium diode.

Forward Bias

Bias means to apply voltage across P-N junction. In Forward bias, anode is connected to positive terminal and cathode to negative terminal of the battery. Fig (1.22) shows forward bias connections and it shows maximum current through the diode. In forward bias the resistance of diode becomes very low. Diode is acting as a closed switch.

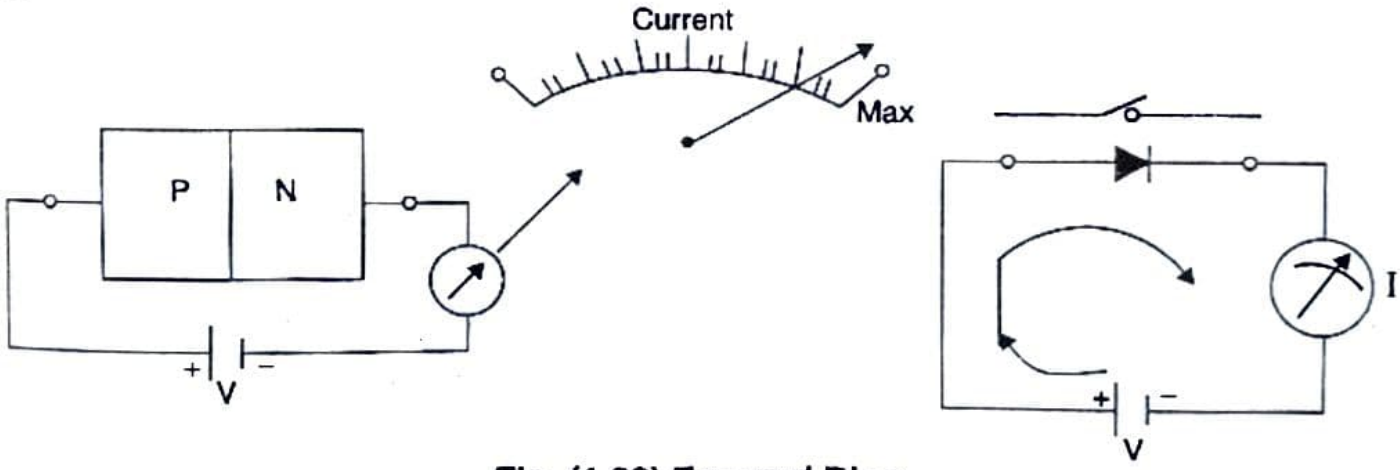


Fig. (1.22) Forward Bias

Let us see how current flows through P-N junction. Negative terminal of the battery will repel electrons at N-region those move towards the junction and they overcome potential barrier because battery voltage is more than 0.7V or 0.3V for Silicon and Germanium respectively. Then electrons move through P-region to the right side of the junction and then collected by positive terminal at the same time holes are attracted by the negative terminal.

Reverse Bias

This is another way of connecting diode, P- region (anode) to negative terminal and N-region (cathode) to positive terminal of the battery, and then it is called as *'reverse bias'*. In this condition since reverse battery voltage is applied across the junction the width of potential barrier increases. This barrier will not allow the movement of majority

carriers from N to P region. The current through diode is not possible which indicates a very high resistance or diode is acting as an open switch as shown in the fig (1.23).

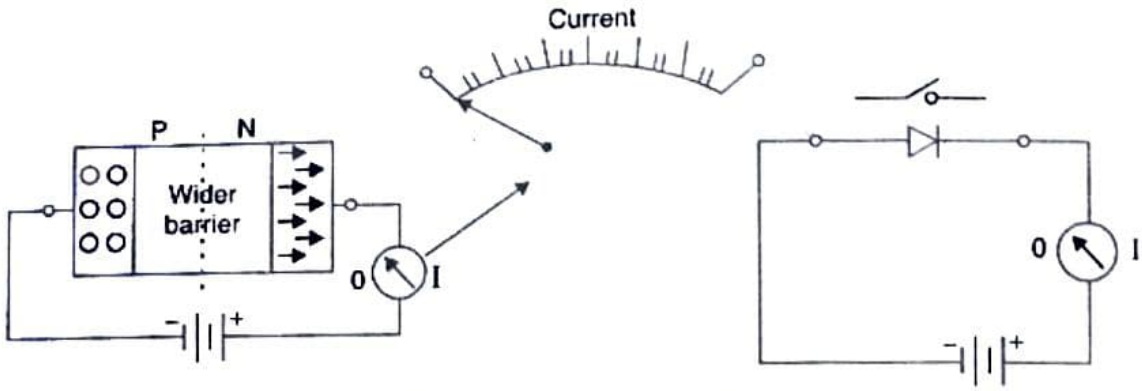


Fig. (1.23) Reverse Bias

Practically, very small current flows through diode, which is called as **'reverse leakage current'** because of minority carries. . P-region has few minority electrons (generated by temperature electron-hole pairs) those are repelled by negative terminal and they flow through the junction towards right side through N-region and then towards positive terminal of the battery.

The above explanation concludes that is current flow through diode is possible only in **one direction**, in forward bias and not in reverse bias.

Applications of Diode

As we have studied P-N junction diode conducts only in one direction, when it is in forward bias. Main application of such diode is to convert bi-directional AC supply into unidirectional DC supply, this process is known as **rectification**. Let us see use of diodes in rectifiers.

Half Wave Rectifier

AC Voltage consists of two half cycles as shown in the fig. (1.24) upper half cycle is known as positive half cycle and lower cycle is known as negative half cycle.

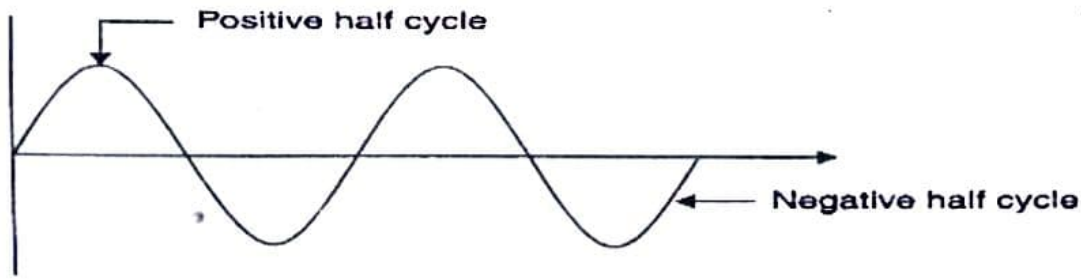


Fig (1.24) AC voltage waveform

Half wave rectifier is a simple and low cost rectifier circuit it is used where high quality DC is not required for example to operate Night lamp, Radio circuit, etc.
As shown in the fig. (1.25) a step down transformer is connected to step down main's AC Supply. A diode is connected in series with load 'R_L' and output DC voltage is taken across R_L.

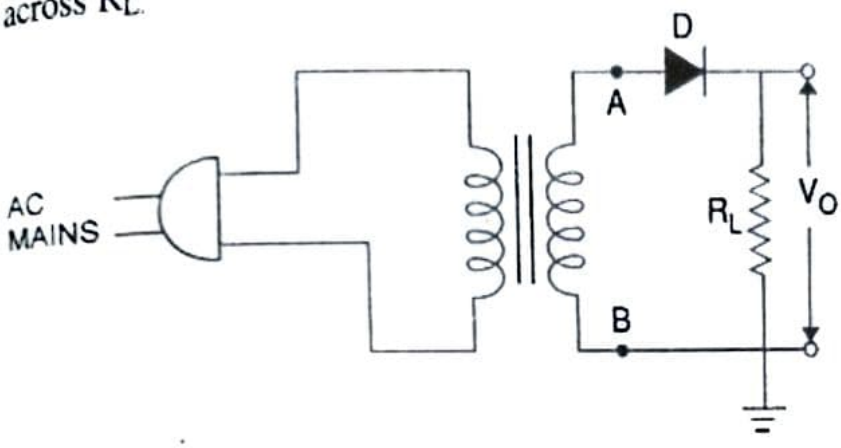


Fig (1.25) Half wave Rectifier

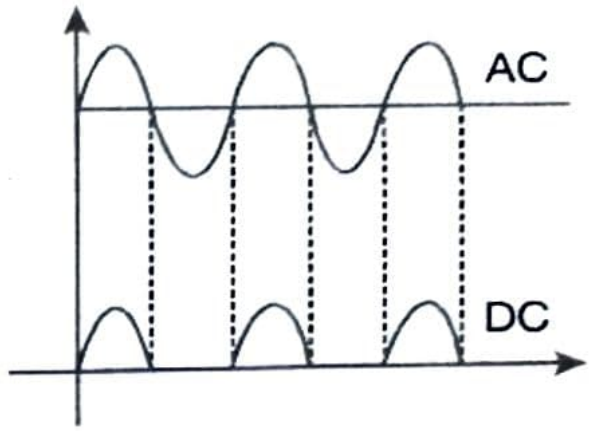


Fig (1.26) Waveforms

Working

Refer its circuit diagram and waveforms.

(i) In First half cycle or positive half cycle of AC voltage, suppose transformer secondary terminal 'A' is positive with respect to terminal 'B' refer fig. (1.27a) diode becomes forward biased it is acting as a closed switch. The current flows through the circuit its flow is from terminal A through R_L to terminal B as shown in fig (1.27a). Thus same output voltage is developed across R_L similar to half cycle of AC input.

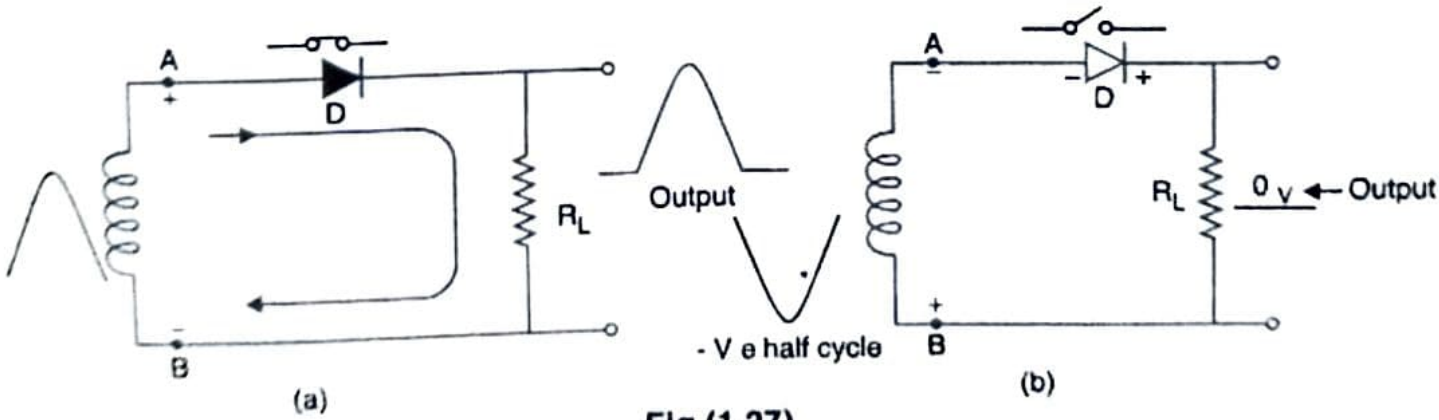


Fig (1.27)

(ii) In the next cycle (negative half cycle) transformer terminal 'B' is positive with respect to terminal 'A' now observe the fig. (1.27b) diode becomes reverse biased, diode is acting as an open switch thus current through circuit is not possible it is blocked by diode. The output voltage

(Vout = I x R_L = 0 x R_L = 0V) is zero refer waveform fig. (1.26).

Thus diode will conduct only at positive half cycle and it rectifies negative half cycle hence it is known as half wave rectifier.

Types of diodes

In addition to rectifier diode there are other kinds of semiconductor diodes, which are manufactured for special applications

1) Zener diode 2) Light emitting diode (LED) 3) Photo-diode
4) Varactor diode 5) Schottkey diode 6) Tunnel diode

Let us see important types, symbols and applications in brief.

1) Zener Diode

Zener diode is designed to get stable voltage and it is operated in **reverse bias** only. By adjusting the doping level, zener diodes of different voltages from 2 to 200V and with different wattage are manufactured. Fig (1.28) shows its operating I-V characteristics and symbol. The Zener gives useful working in reverse bias only in forward bias it conducts just like rectifier diode, it is therefore very important to know about its reverse characteristics.

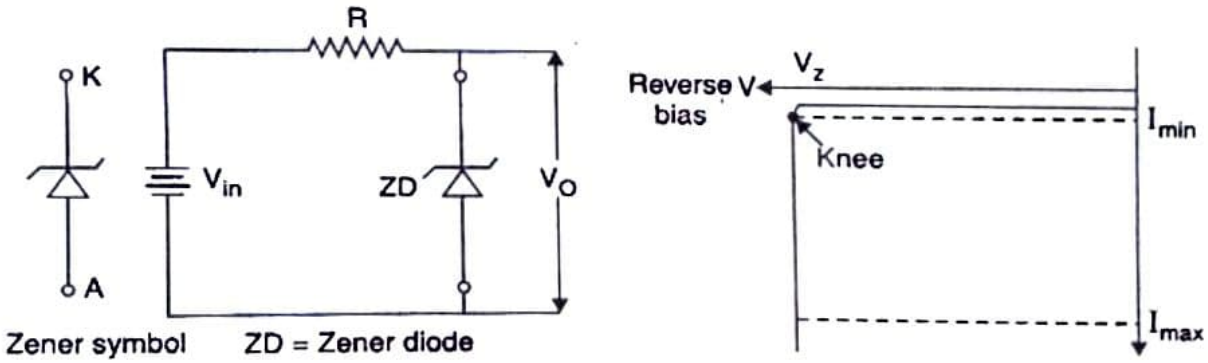


Fig (1.28) Zener Diode and I-V Characteristics

As shown in the fig (1.28) a reverse bias voltage is applied across zener. Initially, as voltage across zener increases the current through zener is very very small and at a particular voltage (V_Z) zener goes into breakdown. At this voltage current through zener suddenly increases but voltage across it remains constant. For further increase in voltage zener keeps voltage across it constant and it draws more current through itself.

Application : Zener diodes are used in DC power supplies as a voltage regulator or in special circuits to stabilize the voltage.

2) Light Emitting Diode (LED)

This diode plays an important role in indicators and displays. LED is constructed with Gallium (Ga), Arsenic (As) and Phosphorus (p) instead of Silicon and Germanium. When light-emitting diode is in forward bias it emits light instead of heat generated by normal diode. When P-N junction is forward bias in case of L.E.D. some of the holes combine with electrons of N-region and some of the electrons from N combine with holes of the P-region. Each recombination radiates light or photons. LED's are available in three colours Red, Green & Yellow.

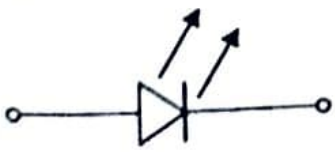
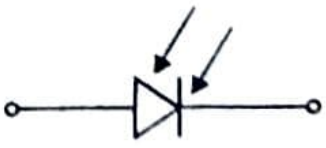
Applications: They are used in 7 segment displays as panel indicators, in watches, calculators, digital meters and as a switch- ON indicators on power supply panels. A series resistance is always required with LED to limit the current because if excess current flows through it, LED may get damaged. (The value of resistance is calculated by ohms law R =V/I)

3) Photo Diode

The function of this diode is opposite to LED, when light is incident on it, it conducts.It is manufactured with cadmium selenide (CdSe), cadmium sulphide (Cds) in addition to Germanium and Silicon. The current through photodiode depends on the light intensity.Photo-diode is normally operated in **reverse bias**. When it is exposed to light due of energy of light,electron- hole pairs are generated therefore current flows through it.

Applications : In computer punch card; light detectors, sound reproduction of film projector and light operated switches.

Comparison

LED	Photo Diode
1. It emits light in when it conducts.	1. It conducts when light is incident on it.
2. It is operated in forward bias	2. It is operated in reverse bias
3. Symbol 	3. Symbol. 
4. Light intensity depends on current through it.	4. Current through it depends on light intensity.
5. Used as indicators or displays.	5. Used as light sensor or transducer.
6. Material used Gallium, Arsenic Phosphorus.	6. Cadmium selenide, Cadmium Sulphide.

1.6 TRANSISTORS

Basically, this name transistor is made up of two terms *"transfer-of-resistor"* because its resistance transfers from one value to other value depending on its bias. In other words the current through transistor is controlled by the bias. Bipolar means both, holes and electrons take part in carrying the current through transistor therefore they are called as **bipolar transistors.**

The transistor also is a P-N device but with two P-N junctions and three terminals
(1) Emitter- that emits electrons (2) Base - which controls this electron flow
(3) Collector- that collects these electrons.
There are two basic types of bipolar transistors
(i) NPN transistor and (ii) PNP transistor. Fig (1.29) shows their structure and symbol.

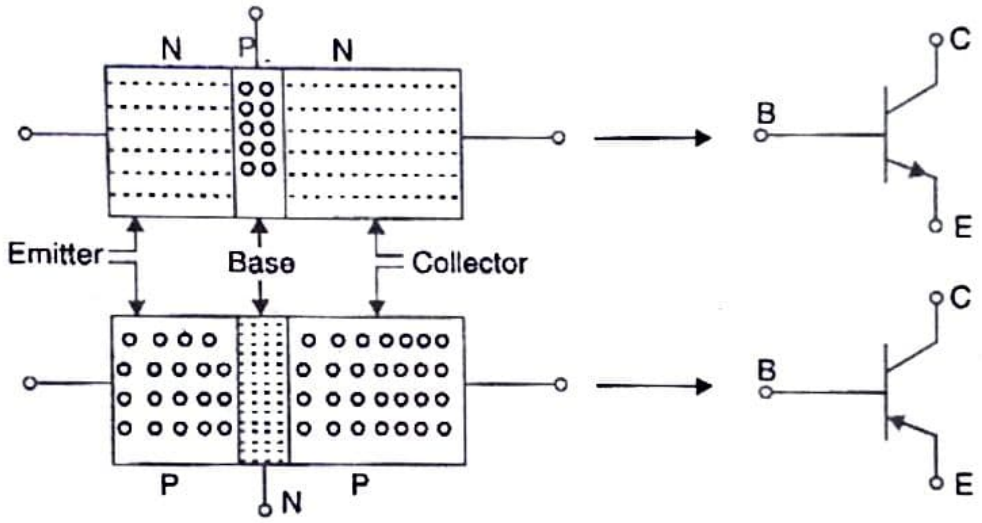


Fig. (1.29) NPN and PNP Transistor

Transistor consists of three semiconductor regions (either Silicon or Germanium) as follows
1) Emitter - It is heavily doped and medium in size.
2) Base - Lightly doped and thin in size.
3) Collector - Medium doped and larger in size.
Thus transistor has two P-N junctions Emitter - Base (EB) and Base - Collector (BC).

Let us see the proper method of biasing a NPN transistor. As soon as transistor is formed, two potential barriers are produced, one is between Emitter and Base and other between Base and collector. To get easy current flow through transistor it must be biased by connecting external DC supply. Hence there must be two batteries to apply bias across E-B junction and B-C junction. We can bias these two junctions in three different ways

(i) FF bias - Both junctions forward biased
(ii) RR bias - Both junctions reverse biased (iii) FR bias - One forward and other reverse biased.

In first (FF) method both junctions are forward biased so large current flows through transistor, transistor behaves as a low resistance, it does not give control action, and therefore this method is not useful. This (FF) condition we get in ***"saturation"*** state. In second (RR) method since both junctions are reverse biased, current through transistor is almost zero transistor behaves as an insulator that is transistor is in ***'cut off'*** which is also not useful state in amplifiers.

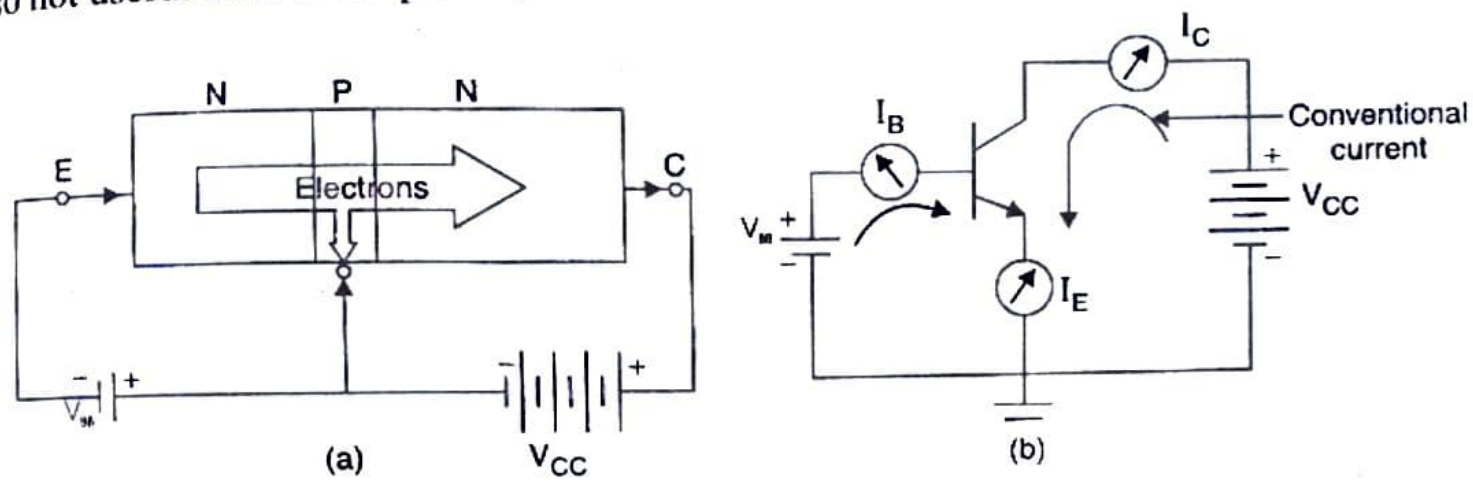


Fig. (1.30) Transistor working

However, the third (FR) method is the correct method in which **Emitter- Base junction is forward biased and Base- Collector junction is reverse biased**. In this method transistor works as a controlled device and transistor is in **Active** region. Refer the fig (1.30) it shows this biasing method. Base-Emitter is biased with low voltage battery (V_{BB}) and Base - Collector with high voltage battery (V_{CC}). Now observe the battery connections, negative terminal of the battery (V_{BB}) it will repel electrons towards the Base, they enter in the Base region. Base is P- type material and it is lightly doped thus few electrons recombine with holes and those number of electrons will pass into the Base from positive terminal (V_{BB}). Rests of the electrons are attracted by positive terminal of the battery (V_{CC}) through Collector.

Collector collects most of the electrons; therefore Emitter current is very close to the Collector current.

$$I_E \equiv I_C$$

But as shown in the fig (1;.30) we can say

Emitter current = Base current + Collector current

$$I_E = I_B + I_C$$

Similarly fig. shows the direction of conventional current from Collector to Emitter.

Alpha and Beta of Transistor

The relation between Emitter current (I_E) and Collector current (I_C) is given by alpha (α) and the relation between Base current (I_B) and Collector current by (β) of the

transistor. 'β' of the transistor tells us that small Base current controls a large Collector current.

$$(\text{Alpha})\alpha = \frac{\text{Collector current}}{\text{Emitter current}}$$

$$\therefore \quad \alpha = \frac{I_C}{I_E} \qquad \text{-------(I)}$$

$$\text{Similarly (Beta) } \beta = \frac{\text{Collector current}}{\text{Base current}}$$

$$\beta = \frac{I_C}{I_B} \quad \text{---(II)}$$

The relation between α and β is given by equation as

$$\therefore \beta = \frac{\alpha}{(1-\alpha)} \quad \text{and} \quad \therefore \beta = \frac{\alpha}{(1-\alpha)}$$

Suppose $I_C = 98$ mA, $I_E = 100$ mA and $I_B = 2$ mA then

$$\alpha = \frac{I_C}{I_E} = \frac{98}{100} = 0.98$$

That means α is always less than 1

$$\beta = \frac{I_C}{I_B} = \frac{98}{2} = 49$$

Where β is greater than 1 and it tells us how much is the smaller Base current than Collector current. β of transistor is important, it is specified by manufacturer it varies from 20 to 400.

Example : If the 'β' of transistor is 49 find the value of α.
Solution : Formula

$$\alpha = \frac{\beta}{(1+\beta)}$$

$$\frac{49}{50} = 0.98$$

Which is true because α is always less than 1.

PNP Transistor

The action of PNP transistor is quite similar to NPN transistor it differs in battery connections only. It requires opposite battery connections. Biasing method is same (FR, refer fig. (1.31) Emitter (P-type) is connected to positive terminal while Base (N-type) to

negative and Collector (P-Type) to negative terminal of high voltage battery V_{CC} or Collector supply. Current through PNP transistor is due to holes as it is due to electrons in NPN transistor.

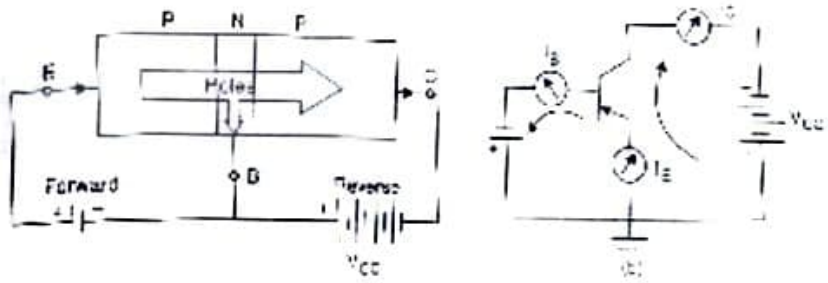


Fig (1.31) PNP Transistor

Here also, we get the similar relations among Base, Emitter and Collector current.

Comparison

P-N Diode	Transistor
1. It has Only one P-N junction	It has Two P-N junctions
2. Diode has two terminals anode and cathode	Transistor is a three terminal device Emitter, base and collector
3. It conducts only in forward bias.	It conducts in FR bias.
4. It is used in rectifier circuits	It can be used as an amplifier or a switch.
Types: Zener diode, L.E.D, Photo diode	Types: NPN, PNP,J-FT, MOSFET

1.7 TRANSISTOR AMPLIFIER

Transistor has two main applications

i) An Amplifier ii) as a Switch

When it is used in amplifier it is operated in active region and avoids cut-off and saturation otherwise it produces distorted signal.

Refer fig. (1.32a) it shows a typical single stage amplifier with voltage divider bias. In the circuit R_1 and R_2 are used to operate the transistor at operating point Q it provides voltage divider bias. Capacitor C_{C_1} and C_{C_2} are called as coupling capacitors they couples AC signal. R_L is used as collector load resistor.

Input signal is given through base, which passes base current. It is amplified in the collector because $I_C = \beta I_B$ Output voltage is taken from the collector w.r.t. ground.

Fig.(1.32b) shows its graphical analysis.

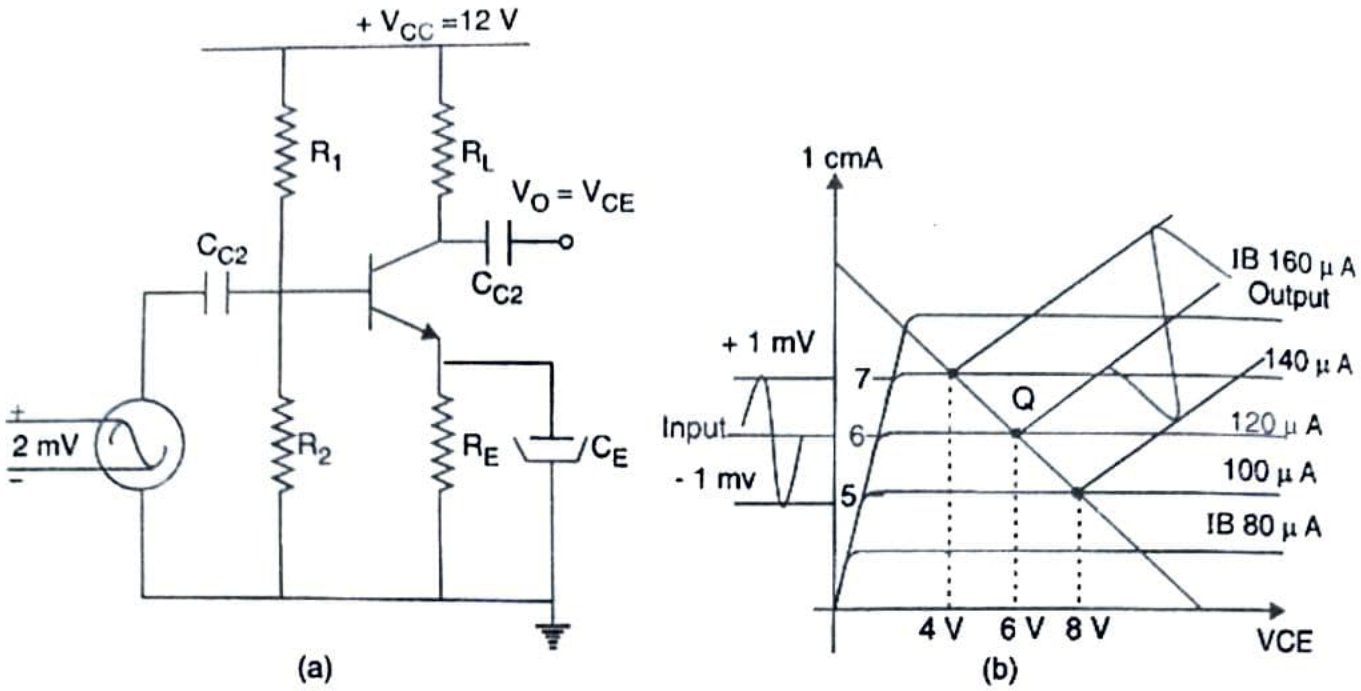


Fig (1.32) Single stage Amplifier

Fig shows input signal of ± 1mV is applied to a transistor amplifier and output is taken from collector to ground ($V_O = V_{CE}$). To understand how signal is amplified by transistor amplifier refer the graph shown by fig (1.32b). It gives the following information

Case 1: When signal is zero or absent

$I_B = 120\ \mu A$, $I_C = 6mA$, (Since $I_C = \beta I_B$ and $\beta = 50$) and $V_{CE} = 6V$

It is known as operating point 'Q' (6V, 6mA)

Case 2: When signal is at peak + 1mV then

$I_B = 140\ \mu A$ $\quad I_C = \beta \times I_B = 50 \times 140 \times 10^{-6} A = 7 \times 10^{-3} A$ $\quad \therefore I_C = 50 \times 140\ \mu A = 7000 \times 10^{-6} A = 7\ mA$

(As I_B increases I_C increases from 6 mA to 7 mA).

Therefore $V_{CE} = 4V$ as input increases output decreases from 6V to 4V because input voltage and output voltage of common emitter amplifier are always 180^0 out of phase.

Case 3: When signal is at peak - 1mV then I_B decreases and I_C also decreases but V_{CE} increases. Refer graph of I_C - V_{CE} as shown in fig (1.32b)

$\therefore I_B = 100\ \mu A$ $\quad \therefore I_C = \beta \times I_B = 50 \times 100 \times 10^{-6} = 5\ mA$ $\quad \therefore V_{CE} = 8V$

These three cases can be summerised as shown in the table.

Input Voltage	I_B	I_C	Output V_{CE}
+1mV	140 µA	7 mA	4 V
0 V	120 µA	6 mA	6 V
-1mV	100 µA	5 mA	8 V

Therefore voltage gain is calculated as

$$\text{Voltage Gain} = \frac{\Delta Vo}{\Delta Vin} = \frac{\text{Change in output voltage}}{\text{Change in input voltage}}$$

$$= \frac{(8V)-(4V)}{(+1)-(-1)mV} = \frac{4V}{2\,mV}$$

$$= \frac{4}{2 \times 10^{-3}}$$

$$= 2 \times 10^{3} \qquad = 2000$$

The voltage gain of amplifier is 2000 that means output signal of amplifier is 2000 times larger than input signal.

1.8 TRANSISTOR AS A SWITCH

When transistor is used as a switch it is operated either in cut-off or in saturation. It avoids active region. In digital circuits like gates in computer circuits transistor is used as an electronic switch. It is operated by applying input voltage (either High or Low) at base terminal.

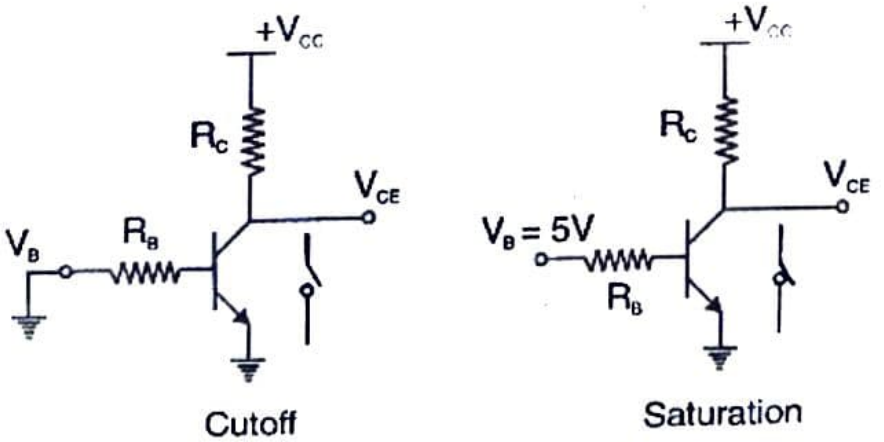


Fig.(1.33) Transistor As A Switch

Case 1: When $V_B = 0$ the base-emitter junction is in reverse bias. The transistor goes in to Cut-off. It is equivalent to open switch as shown.

Case 2: When $V_B = +5V$ the base-emitter junction is in forward bias. The transistor is ON it goes in to saturation. It is equivalent to closed switch as shown.

1.9 SWITCH MODE POWER SUPPLY (SMPS)

SMPS is a modern type of DC regulated power mainly used in computers and television circuits. It consists of a transformer, diode rectifier with capacitor filter but it gives an unregulated DC output. A transistor regulator in which the transistor is operated as a switch hence called as Switch Mode Power Supply (SMPS).

Basic principle

In SMPS instead of making the series pass transistor conducting continuously a transistor is operated in switching mode that acts as a rapidly opening and closing switch. Practically when transistor is operated as a switch it dissipate very small power.

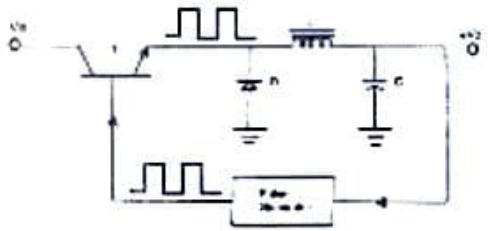
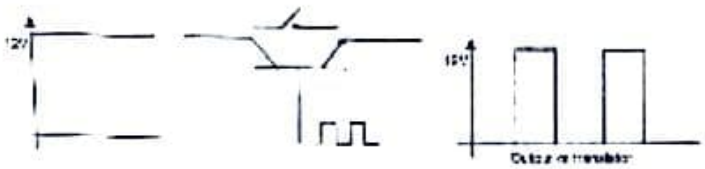


Fig (1.33) Idea of SMPS circuit

Working

As shown in fig. an unregulated DC (Vin) is applied to the switching transistor, which is chopping the voltage in the form of pulses. The switching transistor is operated by a pulse generator operating it at a rapid rate about 20 KHz frequency. When the pulses are connected to the switching transistor its base voltage is rapidly changed between high and low. For instance, when the base voltage is low the transistor is in cut OFF making its emitter low and when the base voltage is high it couples (V_{in}) to its emitter voltage because it acts as a closed switch.

A diode is connected in the circuit to avoid the inductive kickback because back emf is produced by the inductor it may damage the series transistor. An inductor with capacitor form a filter circuit to filter chopped DC into final smooth DC output with a very small ripple.



Advantage of SMPS

(i) Small in size and low cost power supply (ii) Low power dissipation results in less heating. iii)Better regulation and efficiency.

Disadvantage of SMPS

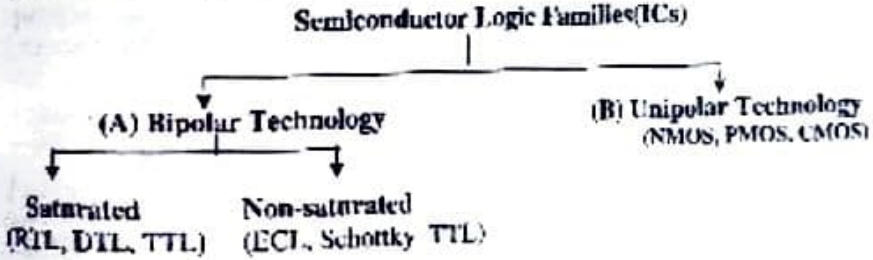
It has maximum advantages as mentioned earlier only two inherent disadvantages.

(i) Since it uses switching technique it generates electromagnetic interference (EMI) like noise spikes, which may affect audio section of the main circuit.

(ii) The control part of the circuit is expensive, complicated and less reliable.

1.10 CLASIFICATION OF ICs

The first semiconductor integrated circuit (IC) was developed in 1958 by J. K. Kilby. Then many different standard semiconductors IC's are developed. These digital IC's can be classified according to the semiconductor device, which is basically used as basic block in manufacturing process of microprocessor and other IC's. These IC's are classified into two groups.



RTL = Resistor Transistor Logic family
TTL = Transistor Transistor Logic family
NMOS = N-channel MOSFET family
CMOS = Complementary MOSFET family
DTL = Diode Transistor Logic family
ECL = Emitter Coupled Logic family
PMOS = P-channel MOSFET family

In bipolar technology, bipolar transistors like NPN transistors are used to construct the logic circuits. In saturated bipolar family the transistors are driven into saturation while in Non-saturated family the transistors are operated slightly below saturation to reduce the propagation delay. In unipolar technology unipolar transistors MOSFETs are used to construct the semiconductor IC's.

Characteristics Of IC's

The performance and the quality of an integrated chip or IC depend on their characteristics. Similarly, the popularity and the cost of semiconductor family depends on various characteristics, the manufacturer specifies these characteristics in the manual

so that user can select that chip for a particular application. There are seven main characteristics as follows

1.Propagation Delay (Speed)
2.Power dissipation
3.Figure of merit
4.Fan-In / Fan-Out

1. Propagation Delay (Speed)

The total time required to execute one instruction or the delay in microprocessor's response for an instruction is decided by this important characteristics. Manufacturer specifies the speed of digital family in terms of propagation delay. The propagation delay is defined as "**the time required to change the output from one logic state to other logic state after input is applied.**" It is denoted by "t_p" for example, TTL logic family has t_p = 10 ns. specifying that it takes 10 nanosec time to change the output from one state to other state.

2. Power Dissipation

Another important characteristic is indirectly related with propagation delay it is power consumption or power dissipation of logic family. It is defined as **"the amount of power in mW dissipated by a gate in an IC"**. Low power dissipation is desirable because it generates less heat and therefore it avoids cooling, power supply and cost problem. Low power dissipation means it should operate with minimum voltage and with minimum current. Manufacturer specifics power dissipation "P_D" in mill watts (mW).

3. Figure Of Merit

This characteristic indicates the total performance or quality of a family. As discussed earlier if resistance of the circuit is increased to reduce the current or power dissipation then the propagation delay increases. In opposite way, if we reduce the propagation delay to make it faster by reducing the resistance then more current is drawn by the gate, which increases the power dissipation.

Therefore both these characteristics, propagation delay and power dissipation are expressed in terms of "figure of merit". It is defined as **"the product of propagation delay and power dissipation"** of the gate. Figure of merit is measured in terms of "PJ" (Picojoules)

(Figure of Merit) $PJ = t_p \times P_D$ $(10^{-9} \times 10^{3} = 10^{-12})$

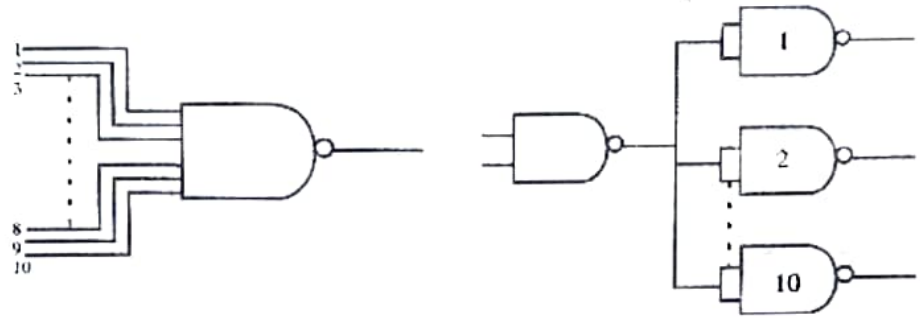
$PJ = \text{nsec} \times \text{mW}$ $(10^{-12} = \text{Pico})$

$= PJ$

For example, in case of (Standard) TTL the propagation delay is 10 nsec and the power dissipation is 10 mWatt therefore its figure of merit is 10 mW x 10 nsec = 100 PJ The lowest figure of merit indicates the best performance.

4. Fan In – Fan Out

Mostly, the capacity of driving the loads of a gate is expressed in terms of fan out. Fan in is the **"number of inputs to a gate".** While fan-out is defined as **"the maximum number of gates (loads) from the same family which can be driven reliably".** Fan out is also known as loading factor. e.g. in case of TTL fan in and fan out is 10 that is maximum 10 TTL gates can be connected to the output terminal of a TTL gate.



Fan In **Fig. (1.34)** **Fan Out**

Comparison of CMOS with TTL Family

CMOS Family	TTL Family
It is an unipolar family.	It is a bi-polar family.
Propagation delay is more. It is 50 nsec.	Propagation delay is very low. It is 10 nsec.
Power dissipation is very low,only 10nW.	Power dissipation is very high. It is 10mW.
Figure of merit is 0.002	Figure of merit is 100
Supply voltage range is 3V to 18V.	Supply voltage range is 4.75V to 5.25V.
Fan-in Fan-out is >50.	Fan-in Fan-out is 10.
Numbering system is 74CXX.	Numbering system is 74XX.

QUESTIONS

1. Select the correct answer

a) A doped semiconductor is known as…….. semiconductor.
   i) Intrinsic ii) extrinsic iii) none of these

b) In …….. semiconductor holes are the majority charge carriers.
   i) N-type ii)P-type iii) both type

c) A P-N diode conducts in …….. bias
   i) forward ii)reverse iii) both

d) The process of converting AC in to DC is known as …..
   i) rectification ii)amplification iii) transformation

e) When transistor is used in amplifiers it is operated in .... Region
i) cut-off ii)saturation iii) active

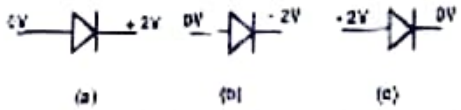
f) In transistor circuits base-emitter junction should be in..... bias
i) forward ii)reverse iii) both

g) In SMPS transistor is operated as a ....
i) rectifier ii)amplifier iii) switch

h) Propagation delay of an IC is measured in.....
i) mSec ii)nSec. iii) Sec.

i) CMOS family is a ..... Family
i) bipolar ii)unipolar iii) none of these

j) In circuit ..... diode is in forward bias.



k) A zener diode when used as a voltage regulator it is .....
i)unbiased ii) in forward bias iii)in reverse biase

l) LED is used as a....
i) rectifier ii) Voltage stabilizer iii) indicator

m) In transistor structure, the region lightly doped having thin area is .....
i) base ii emitter iii) collector

n) If $I_B = 30\ \mu A$ and $I_C = 1.5$ mA then β is.....
i) 0.5 ii) 20 iii) 50

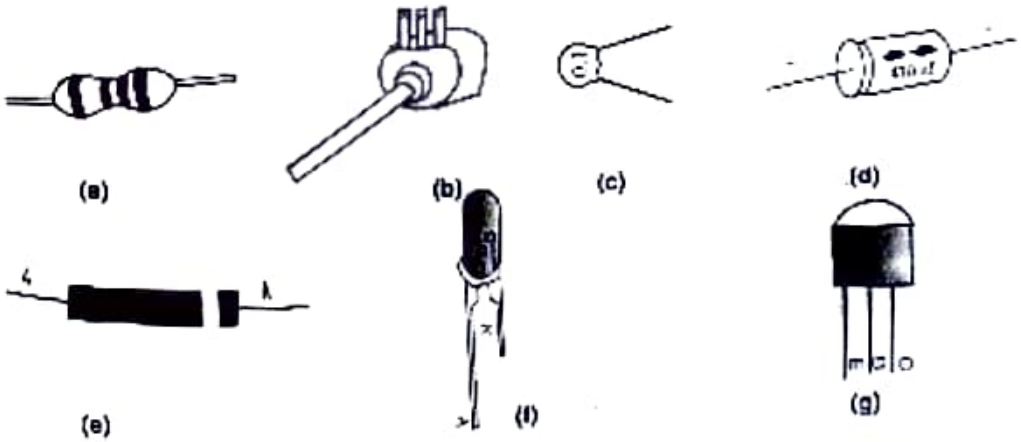
Answer: a) extrinsic b) P-type c) forward d) rectification e) active
f) forward g) switch h) nSec. i) unipolar j)fig. (b)
k) in reverse bias l)indicator m) base n)50

2. State True or false

a) Adding pentavalent impurity forms p-type semiconductor.
b) Photodiode conducts in reverse bias.
c) Transistor can rectify AC supply.
d) In transistor circuits alpha is always less than 1.
e) TTL is a bipolar family.
f) Figure of merit is measured in Joules.
g) SMPS is an unregulated power supply.

Answer: a) False b) True c) False d) True e) true f) False g) false.

3. Write the difference between N type and P-type semiconductor.
4. Draw P-N diode in forward and reverse Bias and explain.
5. Explain The working of half wave rectifier.
6. Compare LED with Photo diode.
7. Explain the construction of transistor and state its types.
8. What is the biasing rule of transistor?
9. Define alpha and Beeta of transistor.
10. Write the difference between Diode and Transistor.
11. Draw a circuit diagram of single stage Amplifier.
12. Explain how transistor is used as a switch.
13. Write advantages and disadvantages of SMPS.
14. Write a short note on SMPS.
15. Classify semiconductor ICs.
16. Define a) Propagation delay b) Fan-in/Fan-out.
17. Compare TTL with CMOS family.
18. In a transistor circuit if emitter current is 10mA and base current is 2mA find collector current, α and β. (Ans:Ic =8mA, $\alpha = 0.8, \beta = 4$)
19. In a transistor circuit $I_C = 12$mA, $I_B = 0.6$mA. Find I_E, α and β. (Ans: I_E=12.6mA, $\alpha = 0.95, \beta = 20$)
20. Identify the following components:



Ans: a) carbon resistor b) Carbon Pot c)ceramic Capacitor d) electrolytic capacitor e) Rectifier Diode f) LED g) Transistor

2. LOGIC GATES & SEQUENTIAL CIRCUITS

INTRODUCTION

The aim of this chapter is introducing the relation between mathematics and logic by means of Boolean algebra and importance of logic gates. The logic gates are the basic circuits of digital electronic systems just like resistors, capacitors in analog circuits. This topic explains details of these gates and examples based on logical equations to get perfection in logical thinking. It also includes Boolean laws and few applications of logic gates.

2.1 LOGIC GATES

The logic gate is an electronic circuit, which has one or many inputs but only one output as shown in fig. (2.1). The output is expressed in terms of binary code 0 or 1.

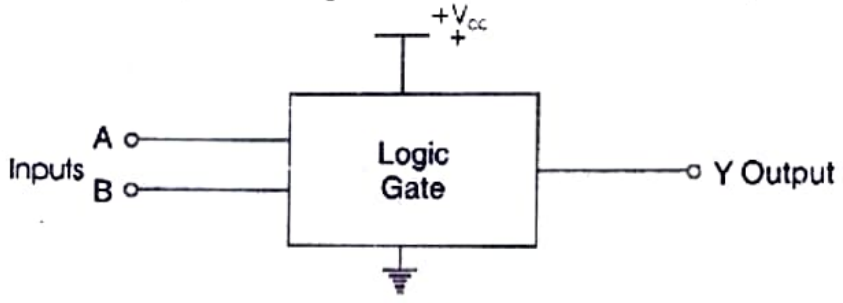


Fig. (2.1)

Logic gates can be constructed by using simple switches, relays, vacuum tubes, transistors, or diodes. Now due to their availability in the form of integrated circuits (ICs) they are used to construct digital circuits. Before studying different types of logic gates let us study few related terms.

(i) Logic - It is the statement, which gives condition for high output. Logic may be in terms of High/Low, True/ False, 1/0 etc.

(ii) Truth Table - It is a table in which logic is expressed by taking all possible combinations of inputs and respective outputs. The number of possible combinations in a truth table is given by 2^n e.g. if there are two inputs (n = 2) then $2^n = 2^2 = 4$ rows, if n=3 then $2^3 = 8$ rows are there.

(iii) Boolean equation - It is a mathematical equation representing particular gate for input and output. Scientist Boolean invented a new kind of algebra; in this algebra he proved that binary logic is valid for letters and symbols instead of words.

(iv) **Symbol** - Each gate is represented by a logical symbol like resistor symbol.
There are six different types of logic gates, out of six, three gates are basic and the other gates are prepared from these basic gates with little modification.

Types of Gates

(A) Basic Gates
1) OR gate 2) AND gate 3) NOT gate / INVERTER

(B) Derived gates
4) NOR gate 5) NAND gate 6) EX-OR gate

Out of these gates the two gates NOR and NAND are called **"universal"** gates because it is possible to construct all other logic gates by using exclusively NOR or NAND gates. Let us study each gate in detail with logic, truth table, Boolean equation, symbol, and simple circuit.

(1) OR gate

This gate is known as **"Inclusive - OR'** gate. Consider A and B are the inputs and Y is the output. The gate is called any or all gate.

Logic - *"If A is true or B is true then Y is true"*

It means that if A and B are the inputs then either A should be high or B should be high or both then output is high where high is represented by binary 1 and low by binary 0.

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Truth Table

Symbol

Boolean equation $Y = A + B$ '+' sign indicates OR function

Simple circuit

Let us construct using A and B switches. For the output a lamp or LED can be used as shown. Consider, Switch open = 0 Switch close = 1
LED ON = 1 LED OFF = 0

A	B	LED (Y)
Open	Open	OFF
Open	Close	ON
Close	Open	ON
Close	Close	ON

Truth Table

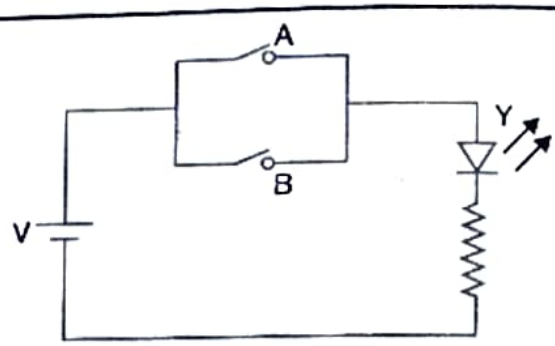


Fig. (2.2) OR gate

Fig (2.2) shows simple OR gate circuit where LED glows when either switch A is closed or B is closed or both because circuit becomes a closed circuit. But when both A and B are open then the circuit becomes open circuit, current cannot flows through the LED thus the LED remains OFF.

(2) AND gate

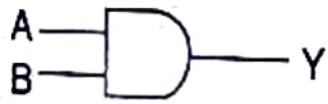
This is another basic logic gate, logic is different than OR logic

Logic - *"If A is true and B is true then Y is true".*

It means that the output of AND gate is high only when both inputs are high.

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table



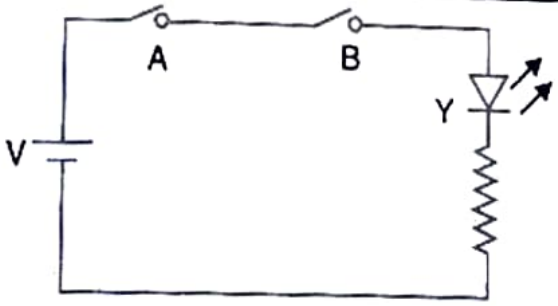
Symbol

Boolean equation $Y = A \bullet B$ or $Y = AB$

If AND gate is three input gate then Y = ABC and so on. Dot indicates an AND operation.

Simple circuit

In this logic switch A and switch B are connected in series so that LED glows only when both are closed otherwise when any one switch is open the circuit becomes an open circuit where current flow is not possible.



A	B	LED (Y)
Open	Open	OFF
Open	Closed	OFF
Closed	Open	OFF
Closed	Closed	ON

Fig. (2.3) AND gate

3) NOT gate

This gate is always single input gate it inverts the input in the output therefore known as "**inverter**".

Logic - *" If A is true then Y is false and if A is false then Y is true".*

A	Y
0	1
1	0

Truth Table

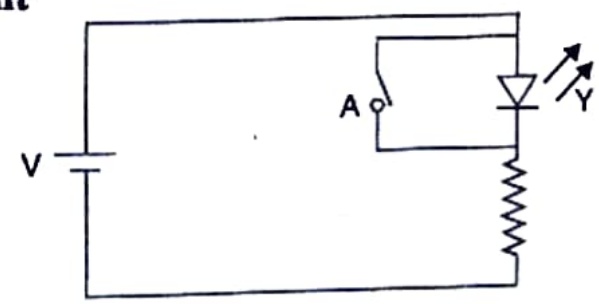
Symbol

Boolean equation

$Y = \overline{A}$ (Y equal to A bar) Bar r epresents inversion.

To get an inverted logic the switch is connected across the lamp. When the switch is closed LED is shorted its anode is directly connected to cathode which will not give forward bias across the LED hence it is OFF. When the switch A is open short circuit across LED is removed current flows through LED hence the LED becomes ON as shown in fig. (2.4).

Simple circuit



A	LED (Y)
Open	ON
Close	OFF

Fig. (2.4) NOT gate

4) NOR gate

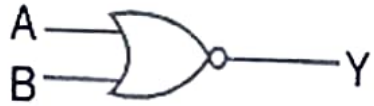
This is the complement of OR gate. If the logic of OR is inverted then it is known as (NOT OR) or NOR gate. The gate is called nothing all gate.

Logic - *" If A is true or B is true then Y is false.'*

This means if any one or both inputs are high then output becomes low.

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

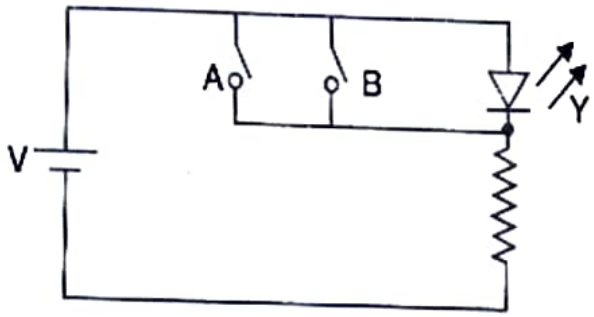
Truth Table



Symbol

Boolean equation $Y = \overline{A + B}$

because '+' sign indicates OR and bar is inversion of OR.



A	B	LED (Y)
Open	Open	ON
Open	Close	OFF
Close	Open	OFF
Close	Close	OFF

Fig. (2.5) NOR gate

Simple Circuit

Refer the circuit of OR and NOT, OR is a parallel circuit of switches but it is in series with lamp. It gets inverted if this parallel switch circuit is connected across the lamp. Because across the lamp means NOT circuit.

5) NAND GATE

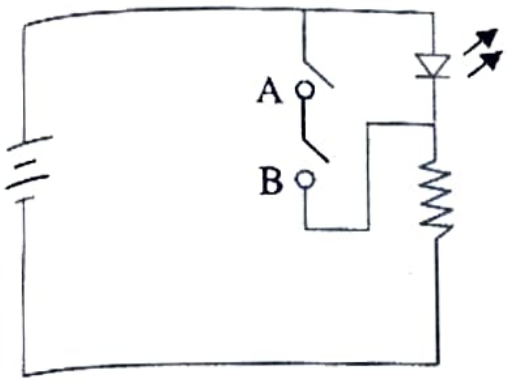
Logic – *"If A is high and B is high then Y is false"*

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Truth Table

Symbol

Boolean equation $Y = \overline{A \bullet B}$ or $Y = \overline{AB}$



A	B	Y
Open	Open	ON
Open	Closed	ON
Closed	Open	ON
Closed	Closed	OFF

Fig. (2.6) NAND gate

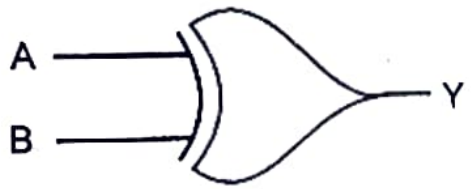
6) EX-OR GATE

The Exclusive OR gate is sometimes referred as *"any but not all"* gate. The term "Exclusive OR" gate is shortened to "XOR". The main difference between an inclusive OR and an exclusive OR is, in EX-OR when both inputs are high the output is low but in inclusive OR, output is high. Now let us see two input EX-OR operation.

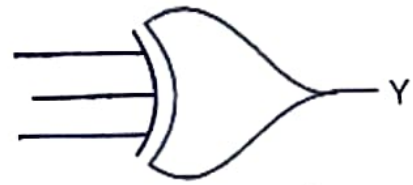
Logic - *"When A is true OR B is true **but not both** then Y is true."*

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Truth table



2 Inputs EX-OR



3 Inputs EX-OR

Symbol

Boolean equation $Y = A \oplus B$

Exclusive OR gate equation can be written as

$$A \oplus B = \overline{A}\ B + A\ \overline{B}$$

This can be proved as

1. When A= 0 and B= 0

$$Y = \overline{A}\,B + A\,\overline{B}$$
$$= 1.0 + 0.1$$
$$= 0 + 0$$
$$= 0$$

2. When A= 0 and B= 1

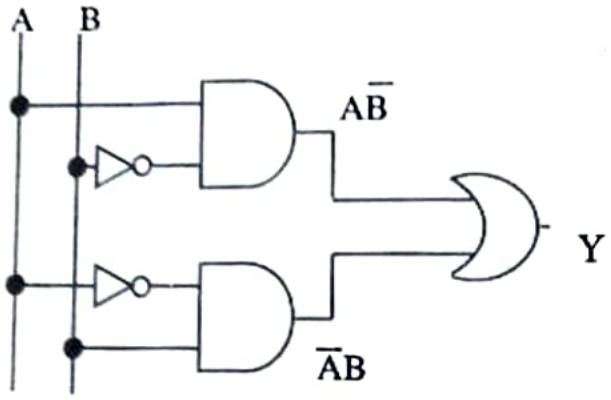
$$Y = \overline{A}\,B + A\,\overline{B}$$
$$= 1\,.1 + 0.0$$
$$= 1 + 0$$
$$= 1$$

3. When A= 1 and B= 0

$Y = \overline{A}\,B + A\,\overline{B}$

$= 0.0 + 1.1$

$= 1$



4. When A= 1 and B= 1

$Y = \overline{A}\,B + A\,\overline{B}$

$= 0.1 + 1.0$

$= 0$

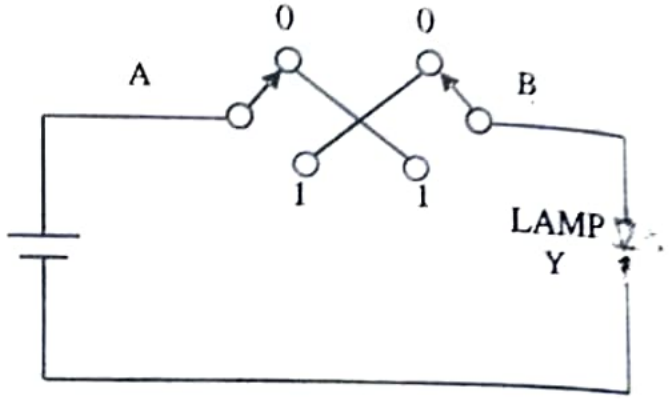


Fig. (2.7) An Exclusive OR gate

Fig. (2.7) illustrates the logic circuit diagram of EX-OR gate using AND, OR and NOT circuit. The equivalent circuit with a SPDT switch is as shown.
The O state and 1 state are shown thus when the inputs are unequal A= 0 & B = 1 and A = 1 & B = 0 the circuit becomes complete, lamp glows and when the inputs are equal A = 1 and B = 1 and A = 0 & B = 0 the circuit is open, lamp does not glow which is the EX-OR operation. Other equivalent circuits can be constructed using the equation $Y = \overline{A}.B + A.\overline{B}$ and connecting an equivalent diode AND, OR and NOT at their place.

ODD / EVEN PARITY

An EX-OR gate shows an interesting property of parity checking. It can be used as a parity checker. As we have already studied the truth table of EX-OR for two inputs but it gives different outputs for three inputs, five inputs i.e. for odd number of inputs. Now let us see three input EX-OR gate with its truth table.

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0*
1	0	0	1
1	0	1	0*
1	1	0	0*
1	1	1	1

Observe the truth table

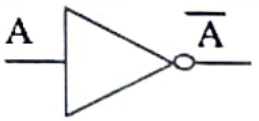
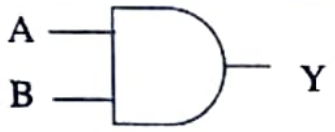
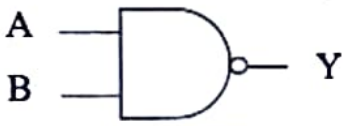
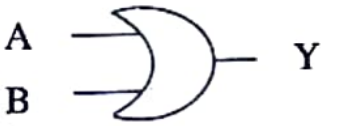
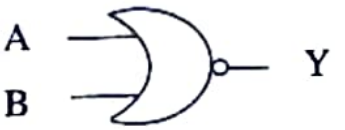
1) When A = 0, B = 1 & C = 1
output is zero because
$Y = A \oplus B \oplus C$
$= 0 \oplus 1 \oplus 1$
$= 1 \oplus 1 = 0$

2) When A = 1, B = 0 & C = 1
$A \oplus B \oplus C = 1 \oplus 0 \oplus 1 = 0$

3) When A=1, B=1 & C = 1
$A \oplus B = 0 \quad 0 \oplus C = 0 \oplus 1 = 1$

The above example shows that when the number of high inputs is even, the output is zero and when the number of high inputs is odd the output is high. Thus EX-OR gate can be used to indicate even / odd parity in given data.

Comparison of Logic gates

Logic function	Symbol	Truth table	Boolean Equation and IC number
NOT	 Input Output	A | $\overline{A}$ 0 | 1 1 | 0	$Y = \overline{A}$ IC 7404
AND	 Input Output	Inputs Outputs A B AND NAND 0 0 0 1 0 1 0 1 1 0 0 1 1 1 1 0	$Y = A.B$ IC 7408
NAND			$Y = \overline{A.B}$ IC 7400
OR	 Input Output	Inputs Outputs A B OR NOR 0 0 0 1 0 1 1 0 1 0 1 0 1 1 1 0	$Y = A + B$ IC 7432
NOR			$Y = \overline{A + B}$ IC 7402
Exclusive OR	Input Output	Inputs Outputs A B EX-OR EX-NOR 0 0 0 1 0 1 1 0 1 0 1 0 1 1 0 1	$Y = A \oplus B$ IC 7486
Exclusive NOR			$Y = \overline{A \oplus B}$

The above table illustrates seven logic functions with this table one can easily remember these gates.

2.2 DE-MORGAN'S THEOREMS

It is frequently required to convert a basic gate into another gate function with the help of inverter. For this purpose De-Morgan has stated two theorems for two universal gates, NAND and NOR. They are known as universal building blocks. i.e. one can change the function of one gate to another by taking the help of an inverter.

First Theorem

This theorem gives an equivalence between NOR and AND gate i.e. we can replace '+' by dot by using this theorem. This theorem can be stated in different ways

"The NOR gate is equivalent to bubbled AND gate."

OR

"The complement of a sum equals the product of complements"

$$\overline{A+B} = \overline{A}.\overline{B}$$

A	B	$\overline{A+B}$	$\overline{A}.\overline{B}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

Truth Table

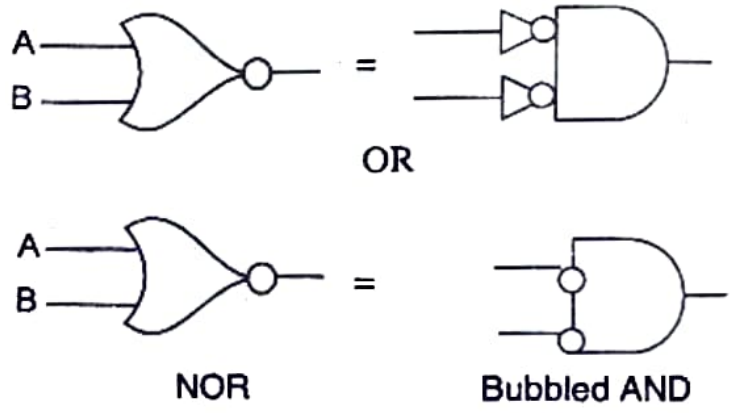


Fig. (2.8)

Fig. (2.8) illustrates the truth table and the symbol of equivalence, bubbled AND whose inputs are inverted and hence by Boolean equation it is represented as $\overline{A}.\overline{B}$ by taking individual bar

Proof $\overline{A+B} = \overline{A}.\overline{B}$

Case 1 A = 0 and B = 0

LHS $\overline{0+0} = 1$

RHS $\overline{0}.\overline{0} = 1$

Case 2 A = 0 and B = 1

LHS $\overline{0+1} = 0$

RHS $\overline{0}.\overline{1} = 0$

Case 3 A = 1 and B = 0

LHS $\overline{1+0} = 0$

RHS $\overline{1}.\overline{0} = 0$

Case 4 A = 1 and B = 1

LHS $\overline{1+1} = 0$

RHS $\overline{1}.\overline{1} = 0$

Second Theorem

This theorem gives an equivalence between NAND and OR i.e. dot can be replaced by '+'. The theorem can be stated as

"The is NAND gate is equivalent to bubbled OR gate."

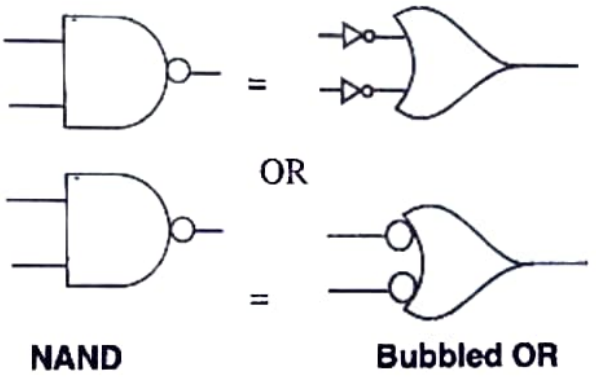
OR

"The complement of product equals the sum of complements."

$$\overline{A.B} = \overline{A} + \overline{B}$$

A	B	$\overline{A.B}$	$\overline{A}+\overline{B}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

Truth Table



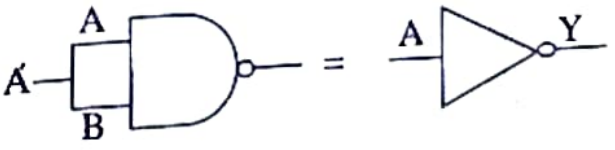
Considering the four cases as explained in the first theorem try to prove this theorem. Verify the truth table.

2.3 NAND GATE IS AN UNIVERSAL BUILDING BLOCK

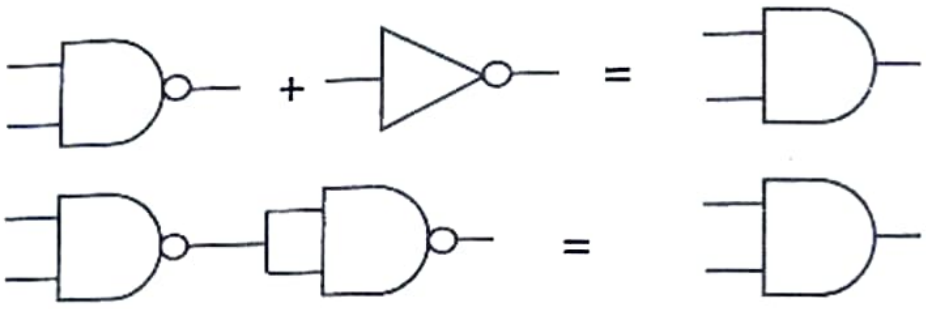
The NAND gate has an interesting property. It can be used to build an OR gate, AND gate or NOT gate. We can construct all other gates using one or many NAND gates, hence the circuit which has a number of gates AND,OR,NOR etc. can be converted into a simple circuit which will contain only NAND. In the next topic you will find that a flip-flop circuit can also be constructed by NAND. In the manufacturers literature you will find NAND gate seems to be more widely available than other gates. Let us construct all other gates using NAND only. (* Follow the same sequence)

Constructions

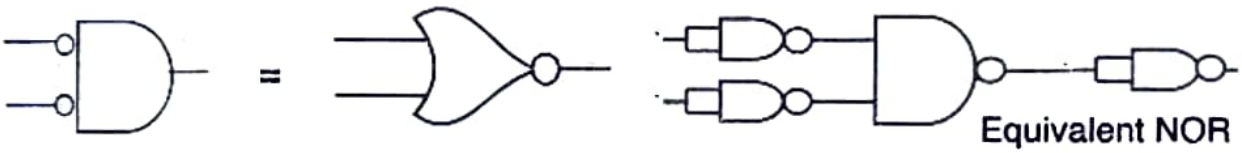
1. **NOT using NAND :** To convert NAND to NOT, refer the truth table of NAND gate. When the inputs are A = 0 and B = 0, the output Y is 1 and when A = 1 and B = 1, the output Y = 0 and thus we can short A and B inputs and make it single input A'.



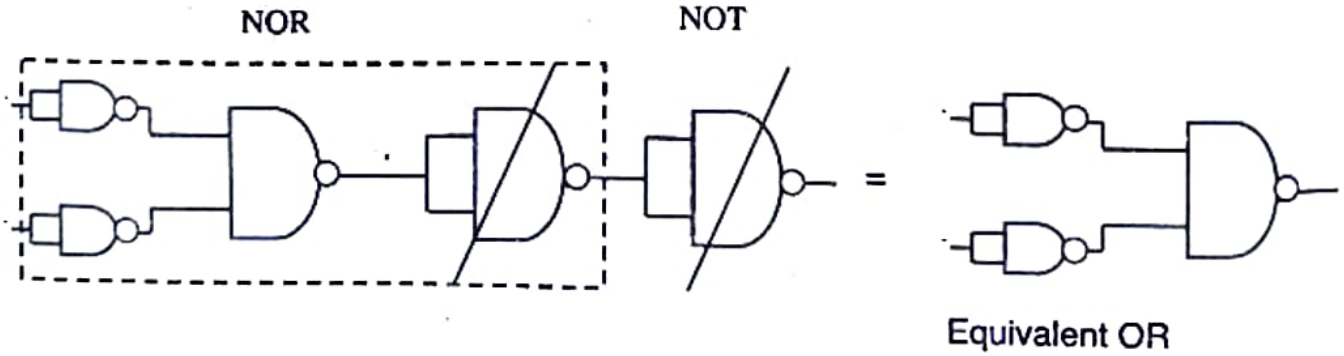
2. **AND using NAND :** Here we can use Boolean equation $\overline{\overline{A.B}} = A.B$
Now you can easily construct an AND gate by just inverting NAND output.



3. **NOR using NAND :** De-Morgan's theorem states that a bubbled AND is NOR. We know NAND equivalent bubble (NOT), and equivalent AND circuit connect them.



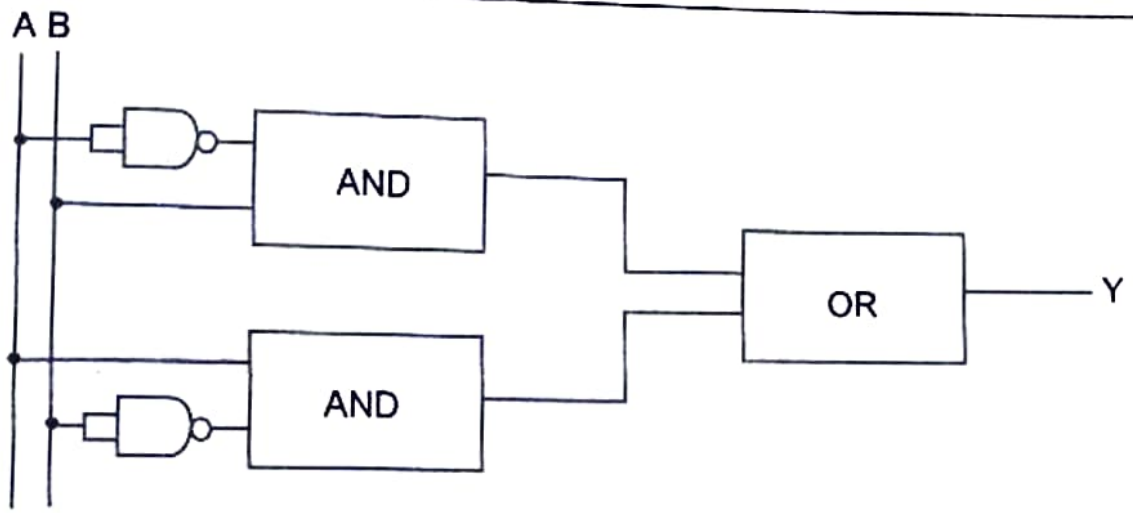
4. **OR using NAND :** Now you can invert the output of NOR $\overline{\overline{A+B}} = A+B$
Fig. shows the construction of an OR using NAND, double inversion gets cancelled



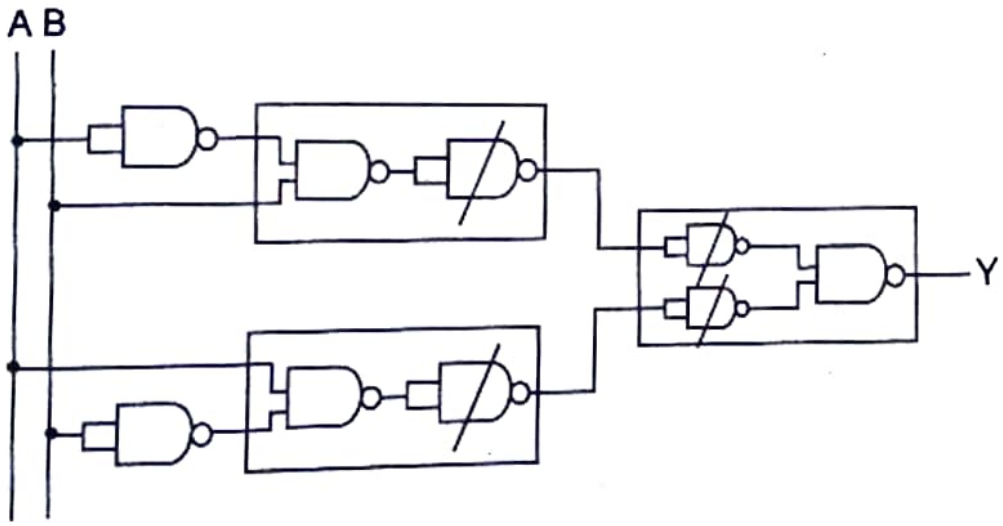
5. **EX-OR using NAND :**
This can be simplified by connecting the circuit $Y = \overline{A}.B + A.\overline{B}$

$= [\overline{A}B] + [A\overline{B}]$

$=$ AND + AND



Substituting AND, OR circuits in the above diagram



Cancel the inverters in series as shown; the rest of the circuit gives an EX-OR circuit.

2.4 HALF ADDER AND FULL ADDER

In computer circuit arithmetic section has different circuits like adder, subtractor, multiplier, integrator etc. The gate circuit performs all these mathematical operations. Here let us design and study a basic circuit of a two-nibble adder circuit. A Four-bit addition requires one half adder for two bit addition of LSB column and three full adders for next column six-bit addition.

Where 1 binary digit = 1 bit, 4 bits = 1 Nibble and 1 byte = 8 Bits

Half Adder: It can add two bits, suitable only for LSB column addition. We know the rules of two-bit addition as shown in the truth table. Now the circuit can be designed by observing the truth table, the carry output and the sum output. Refer the sum outputs, this

is nothing but an EX-OR and refer the carry outputs, we can connect AND gate, the circuit diagram is shown in figure (2.10).

Boolean equation

i) CARRY = A . B ii) SUM = A ⊕ B

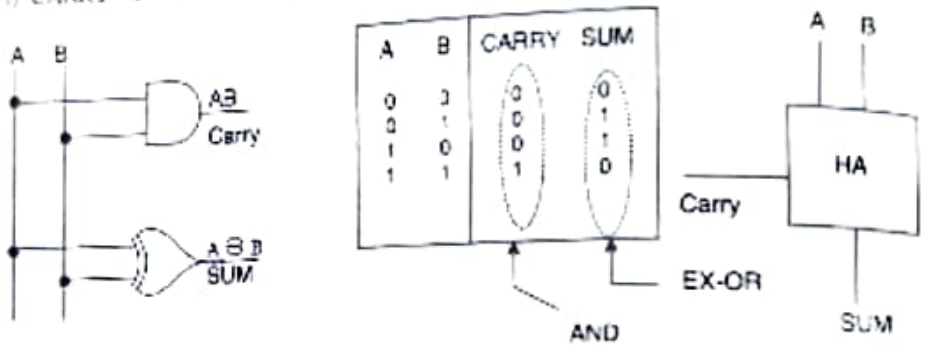


Fig. (2.10) Half Adder

Full Adder : In the higher columns, there may be a carry from LSB column or previous column. Three-bit addition is required which full adder can perform. The full adder circuit can be designed with the help of truth table. As shown by dotted line the output 'carry' is high when AB, BC, CA and ABC are 1. For these combinations we can connect three AND gates and for 'sum' output we can connect three input EX-OR gate.

A	B	C	CARRY	SUM
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

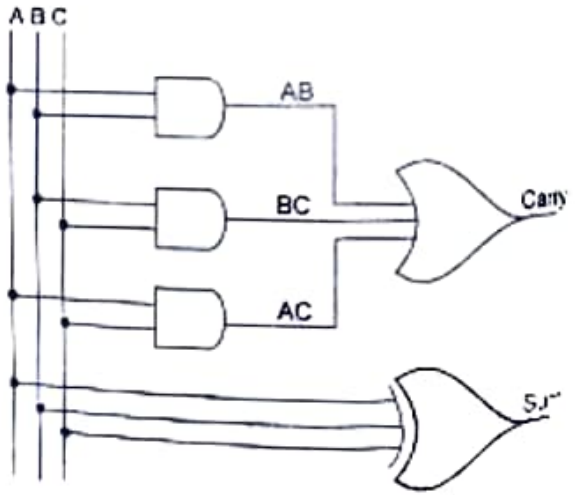
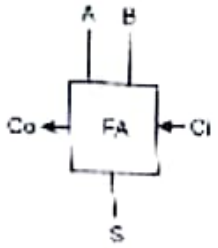


Fig. (2.11) Full Adder

Boolean equations i) CARRY = AB + BC + AC (Refer dotted circles in truth table)
ii) SUM = A ⊕ B ⊕ C

Symbol of full adder :



Binary Adder (Block Diagram)

The two 4 bit number addition can be performed by connecting one half adder and three full adders, as illustrated in fig.(2.12).

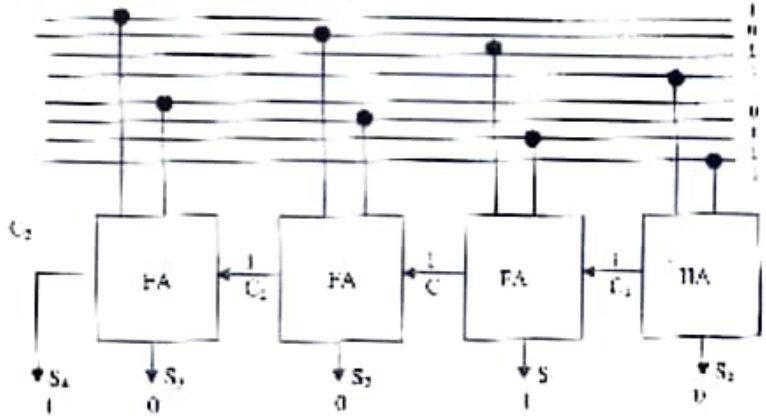


Fig. (2.12)

Consider the two numbers

$A_3\ A_2\ A_1\ A_0$
$+\ B_3\ B_2\ B_1\ B_0$
$C_3\ C_2\ C_1\ C_0$

$S_4\ S_3\ S_2\ S_1\ S_0$

1 0 1 1
+ 0 1 1 1

1 0 0 1 0

Ex-OR inverter

Consider an EX-OR gate, if its one input is kept high and the other input is made variable then it works as an inverter. Consider A = 0 then Y = 1 and when A = 1 then Y = 0 as

shown in the figure. As shown in figure, four EX-OR gates can be used to find one's complement of the input (ABCD) by making a common high input.

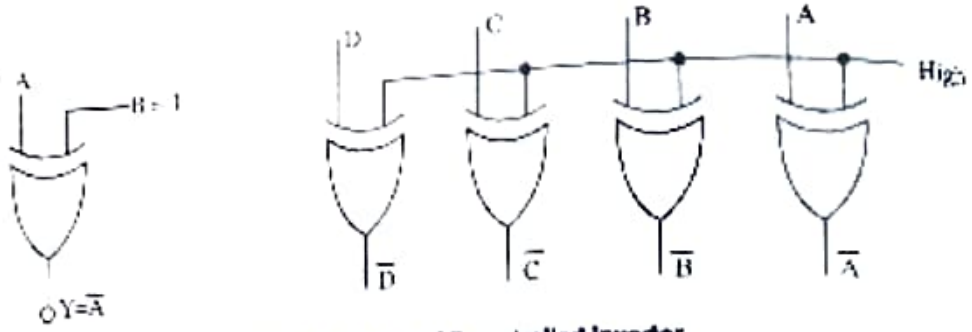


Fig. (2.13) EX – OR controlled inverter

Note that when the common signal is low to this circuit then the EX-OR circuit output does not change it is ABCD only. The common line can be used as a control so that when the line is low it sends ABCD and when it is high it sends A B C D

ADDER/SUBTRACTOR

Let us connect a circuit with EX-OR gates to perform addition as well as subtraction by using a common hardware. Fig. (2.14) shows a 2's complement subtraction by using EX-OR inverter circuit by using a control signal "SUB".

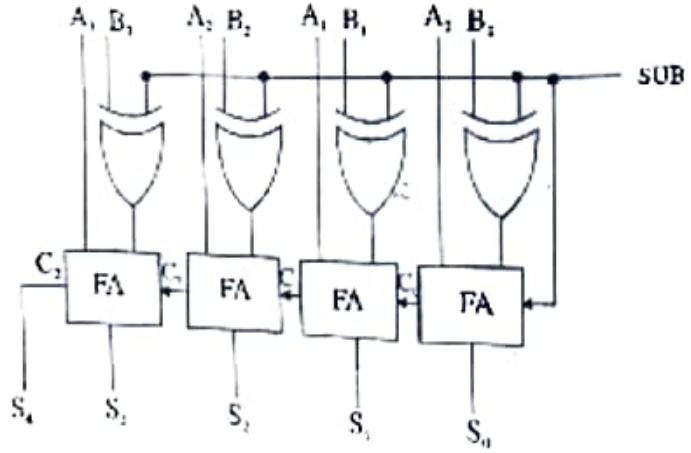


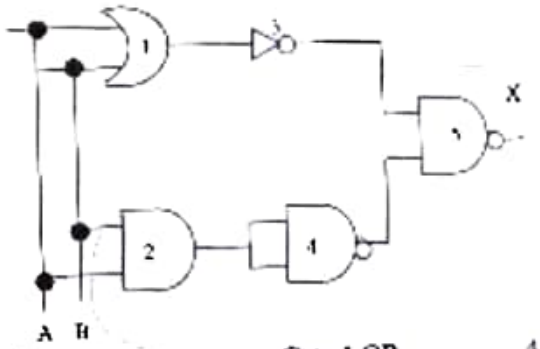
Fig. (2.14) Adder/Subtractor

i) When SUB = 0 the circuit performs binary addition by usual method $(A_3,A_2,A_1,A_0 + B_3,B_2,B_1,B_0)$. The EX-OR circuit sends data without any change.

ii) When SUB = 1 the circuit performs subtraction by 2's complement method. As shown in fig (2.14) EX-OR circuit generates 2's complement of B_3,B_2,B_1,B_0 and performs subtraction. Refer the circuit, the EX-OR circuit will pass the 1's complement of B_3,B_2,B_1,B_0 and 1 is added to the first full adder when SUB is high.

SOLVED PROBLEMS

1. Write the Boolean equation for output X and write the truth table for it. Name gates used.



Solution: As shown in the diagrams try to write output of each gate. Equation is

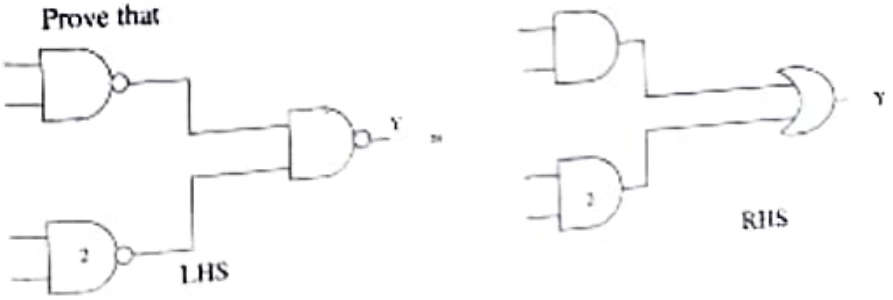
$$\therefore X = \overline{\overline{(A+B)}.\overline{(A\,B)}}$$
$$= (A+B)+(A+B)$$

Truth Table

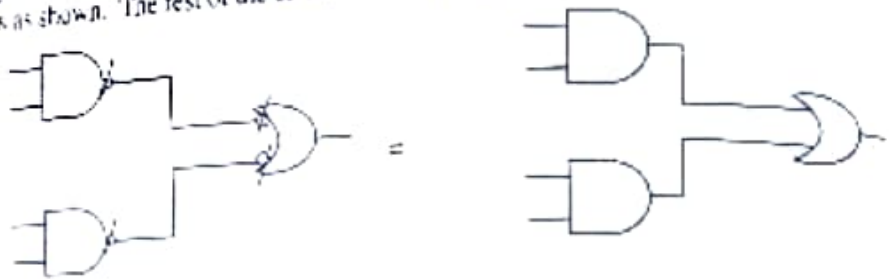
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Gate 1 OR
2 AND
3 NOT
4 NAND inverter
5 NAND

2. Prove that



Solution : Consider LHS the third NAND can be replaced by bubbled OR by applying De Morgan's theorem and then it can be simplified by eliminating two bubbles (inverters) in series as shown. The rest of the circuit results in RHS.

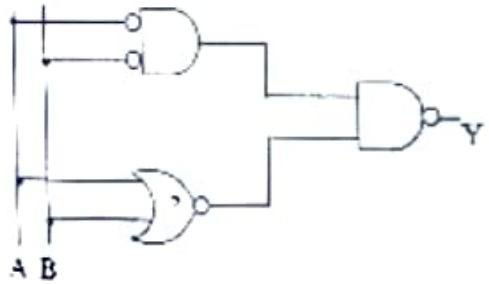


3. Draw the logic diagram using gates for the following equation

$Y = \overline{(\overline{A}.\overline{B})} \ . \ \overline{(A+\overline{B})}$

Solution : Simplifying the equation $Y = \overline{(S)} \ . \ (P)$

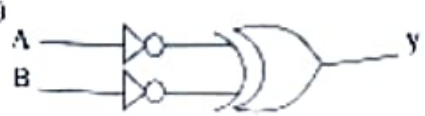
Where $S = (\overline{A}.\overline{B})$ and $P = \overline{(A+\overline{B})}$



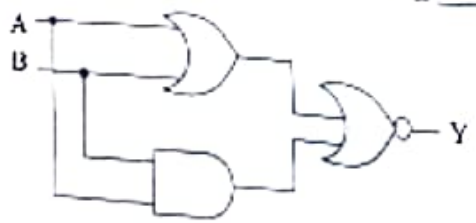
4. Find the Boolean equation for the given logic diagrams

i)

ii)



iii)



Solution : (i) $Y = \overline{A}.B$ (ii) $Y = \overline{A} \oplus \overline{B}$ (iii) $Y = \overline{(A+B)+(A.B)}$

QUESTIONS

1. Select the correct alternatives

i) The output of the NOR gate is high when.....

a) all its inputs are high b) all its inputs are low

c) any one of its inputs is high d) any one of its inputs is low.

ii) It is desired to have output 0 when the two inputs of an electronic gate, are equal. The gate one must use will be.....

a) NAND b) AND c) Inclusive OR d) Exclusive OR

iii) If the input to EX-OR gate is A= 1 and B= X then Y is.....

a) 0 b) $\overline{X}$ c) X

iv) When all inputs are low only then o/p is low this is the logic of

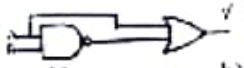
a)OR gate b) AND gate c)EX-OR gate

v) The half adder is a logic circuit which can add ..... bits

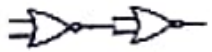
a) 2 b) 3 c) 4 d) 4

vi) In ..... if A =1, B = C then output Y = C

a)NOR gate b) OR gate c)EX-OR gate d) AND gate

vii)  in the given circuit if A=1 and B=1 then y =....

a)1 b)0 c) 0 or 1

viii)  represents ....gate

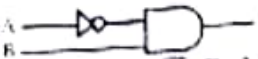
a) OR b) NOR c) NOT

ix) How many NAND gates are required to form AND gate

a) 1 b) 3 c) 2

x) the Boolean equation for the given circuit is....

a) $Y = A \oplus B$ b) $Y = \overline{A} \oplus \overline{B}$ c) $Y = \overline{A \oplus B}$

xi)  the Boolean equation for the given circuit is....

a) $Y = \overline{A}.B$ b) $Y = \overline{A}.B$ c) $Y = \overline{A.B}$

xii) The output of the given logic circuit is

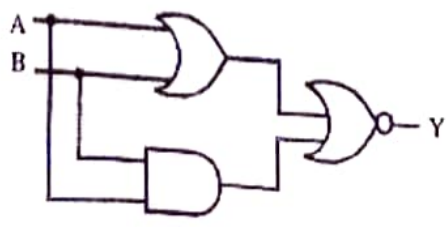
a) 1 b) 0 c) A

Ans:

i) if all inputs are low ii) EX-OR iii) $\overline{X}$ iv) OR Gate v) 2

vi) AND Gate vii) 1 viii) OR ix) 2 x) $Y = \overline{A} \oplus \overline{B}$ xi) $Y = \overline{A}.B$ xii) 1

2. Explain the basic gates with definition, symbol and truth table.
3. Write the truth table for NOT, NOR and AND gate.
4. Construct an AND, NOT and NOR gate by using NAND gate only
5. Draw the logical diagram for the following expression $Y = (A + B) \oplus (A.B)$ where A and B are inputs and Y is the output.
6. How inclusive OR gate differs from Exclusive OR gate. Draw symbols and write truth tables.
7. Explain half adder with its truth table.
8. Explain odd/even parity checking by EX-OR gate.
9. State and prove De-Morgan's theorems.
10. Give the difference between half-adder and full adder. Draw the diagram of full-adder. Write their truth tables.
11. Why NAND gate is called as universal building block. Explain with constructions
12. Draw a circuit diagram of 4-bit adder/subtractor circuit and explain its working.
13. Prove that a) $\overline{A}.\overline{B} = \overline{A + B}$ b) $\overline{A} + \overline{B} = \overline{A.B}$
14. Draw the circuit diagram of controlled inverter and explain.
15. Write the truth table of OR gate using negative logic.
16. Find Boolean equation and Truth table for given circuit.



COMBINATIONAL CIRCUITS

After studying the logic gates, their truth tables and Boolean equations one can understand how these basic digital components are useful while designing different logic circuits like half adder. In addition to simple logic circuits there is a family of some other combinational logic circuits such as multiplexer, demultiplexer, encoder etc. This topic explains these logic circuits by combining different logic circuits hence called combinational logic circuits. These combinational circuits are now available in the form of integrated circuits reducing the size and cost of the system. Basically digital circuits can be grouped in to two - combinational circuits and sequential circuits.

2.5 MULTIPLEXERS

The multiplexer is a typical combinational circuit widely used to connect many sources to one common line to make the system simple, economic and using a single line instead of using separate line for each input source. Multiplexing is the process of connecting many inputs to one output it is a **"many into one"** process with appropriate logic control signals. Multiplexer is also known as data selector circuit since it accepts more than one inputs and connects only one of them at a time to the output.

Multiplexer is a basic block or basic logic combinational circuit with

i) 'n' input data lines .ii)'m' control lines / selector lines

iii) one output line 'Y'

Fig (2.15) illustrates its analogy with a rotary switch to connect its pole to 'n' inputs and a block diagram of a multiplexer with required lines.

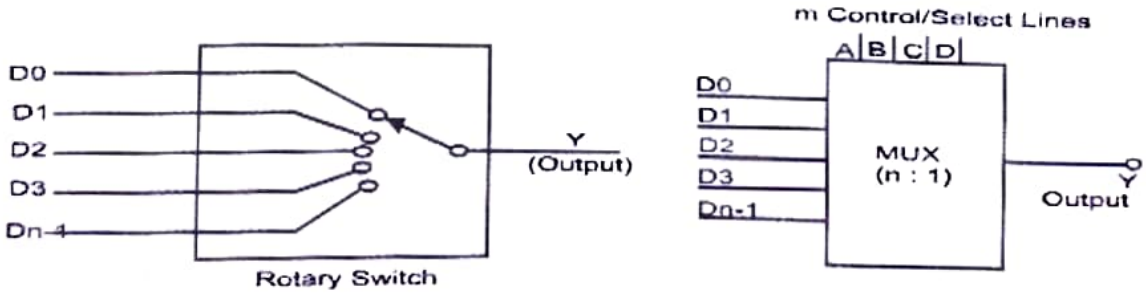


Fig.(2.15) Concept of Multiplexing

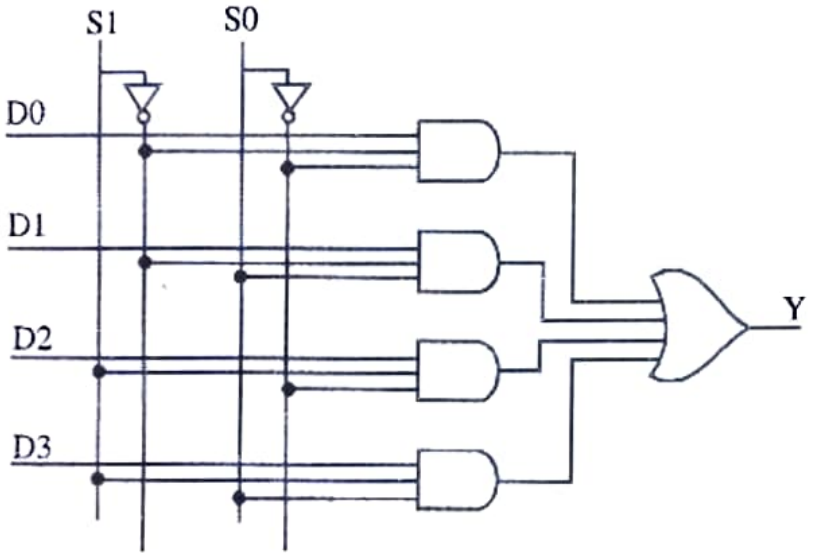
Consider a rotary switch as shown in fig (2.15) its output can be connected to either of 'n' number of inputs just by selecting proper position, it is a manual switch we need electronic circuit and it is multiplexer simply called as "MUX". It has 'n' input data lines (D_0-D_{n-1}) and 'm' (S_3 S_2 S_1 S_0) control lines. For selecting one of the 'n' inputs to the output of MUX it requires minimum 'm' lines 'm' can be calculated by using an equation.

$$2^m = n \text{ .....(formula)}$$

For instance, to design a 4:1 multiplexer 4 input lines 2 control lines are required because $n = 4$, $2^m = 4$, $m = 2$. Let us see typical MUX circuits to understand its working and applications.

i) 4: 1 Multiplexer

As stated earlier, it has 4 input lines (D_0, D_1, D_2, D_3) two control lines (S_1 S_0) (called select lines) and one output denoted by 'Y'. It is very similar to a single pole 4-way switch the pole can be connected to any of these 4 inputs with control signal of selecting the position of the switch. Refer fig (2.15).



S_1	S_0	Y=
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Truth Table

Fig.(2.16) 4 to 1 Multiplexer

Fig. (2.16) also shows a similar arrangement but using an electronic circuit; it is a combination of logic gates but designed to select any one of the 4 inputs and not operated manually.

Working:

i) When the control word is 0 0 (S_1=0 and S_0=0) it selects the first AND gate with input line D_0 refer figure and the first AND gate connections, the input to this gate is D_0,1,1. D_0 is the first input, inverted S_1 as the second and inverted S_0 is the third input. If you trace other AND gates control signal to these AND will make them disable. The final OR gate will connect the output of the first AND gate to the output (Y). Logically it can be stated as when S_1 S_0 = 00 then $Y = D_0$

ii) When (S_1=0 and S_0=1) trace the circuit the second AND gate is enabled and it will connect D1 to the output. This is logically stated as when S_1 S_0 = 01 then $Y = D_1$

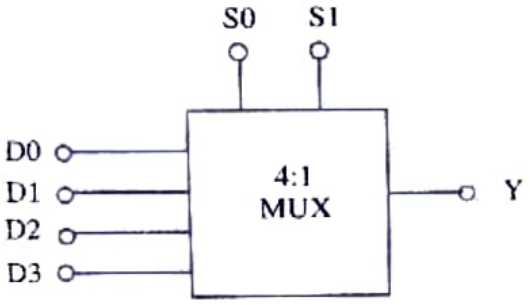
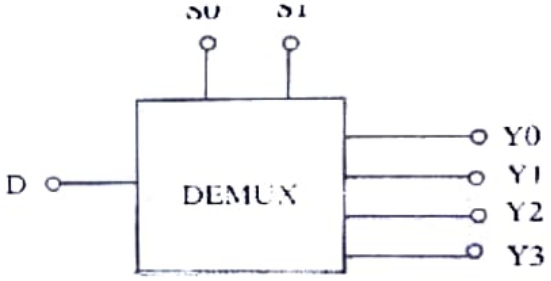
iii) When control word S_1 S_0 = 10 then $Y = D_2$

iv) In this way when S_1 S_0 = 11 then $Y = D_3$

Note that at any instant only one AND is enabled allowing only one data line transmission to the common output line.

2.6 DEMULTIPLEXERS

De-multiplexing is an opposite process to a multiplexing process it performs "**one to many**" operation. It has only one input (D) and 'n' number of outputs (Y0,Y2,Y3..Yn-1) Demultiplexer accepts one input and can distribute it to several outputs the select code or control word determines to which output the input is connected. Demultiplexer can also be used as a decoder e.g. binary to decimal decoder.

Multiplexer	Demultiplexer
i) It is known as MUX	i)It is DEMUX
ii) Multiplexing means many to one	ii)It means one to many
iii) Logic Symbol 	iii) Logic Symbol 
iv) It cannot be used as a decoder. e.g. IC 74153, 74157	iv) It can be used as a decoder e.g. IC 74156, 74155
v) It requires OR gate in the O/P	v) Since it has many outputs OR gate is absent

1 TO 4 Demultiplexer

1 to 4 DEMUX has 1 input (D), which can be connected to four outputs (Y_0, Y1, Y2, Y3) but one at a time. The output is selected by a control word or code present on the select lines as explained earlier since n = 4 and $2^m = 4$, the number of select lines must be two S0 and S1. Fig (2.17) shows its logic symbol and circuit diagram.

Working

Select logically the data input (D) is to be connected either of four outputs. The desired output is decided by control word given by means of select lines. When both S0 and S1 are low (S0=0 and S1=0) it will make S0=1 and S1=1 and this will make the first AND gate enable.

The data input will appear on the output Y0. In this way
Y1 can be selected by making S1=0 and S0=1,
Y2 can be selected by making S1=1 and S0=0,
Y3 can be selected by making S1=1 and S0=1.

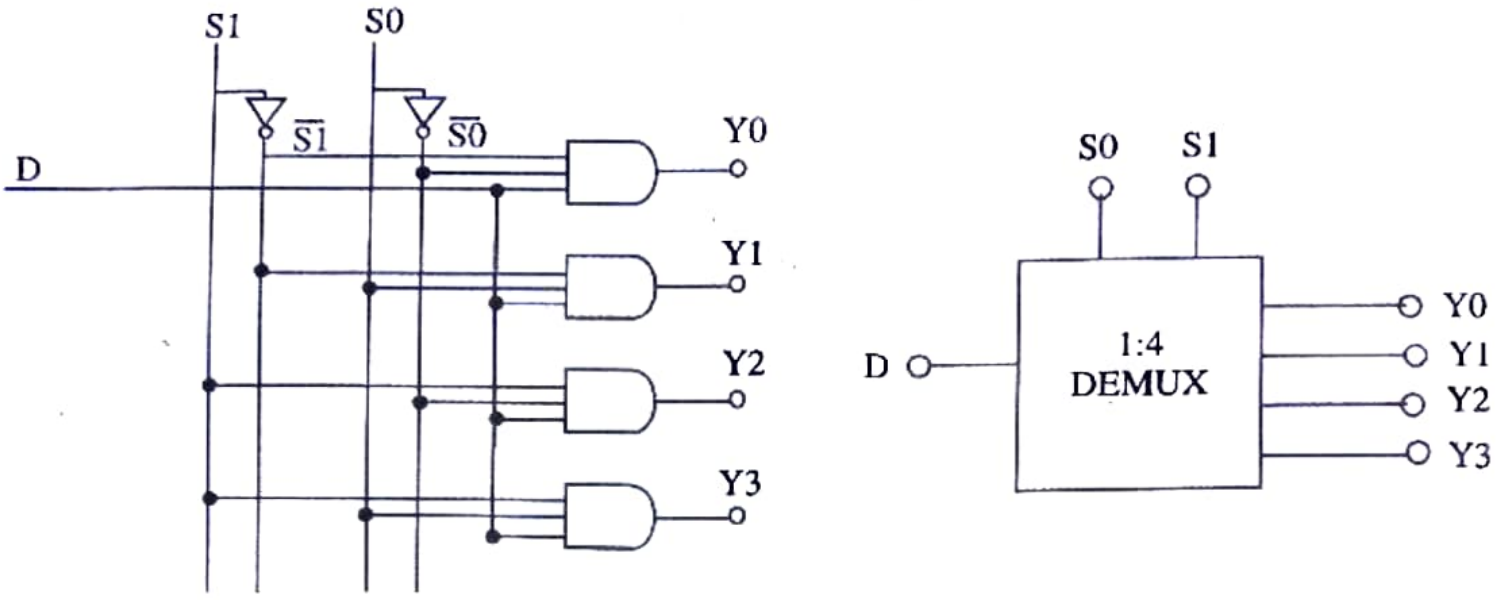


Fig.(2.17) 1 to 4 De-multiplexer

Truth table

S_1	S_0	D=
0	0	Y_0
0	1	Y_1
1	0	Y_2
1	1	Y_3

Now you can try to draw the circuits of 1 to 8 and 1 to 16 Demultiplexer.

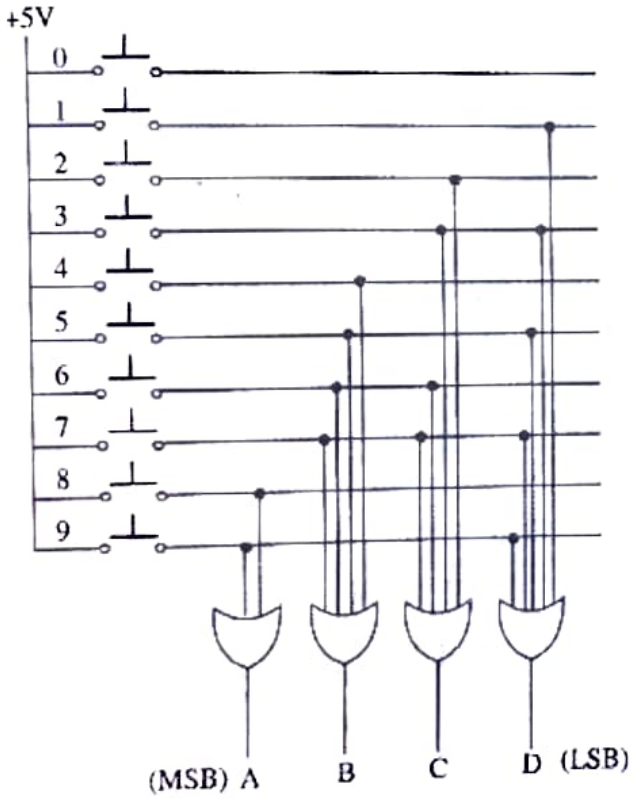
2.6 ENCODERS

A digital circuit like calculator is having input numbers from 0 to 9 expressed as common inputs but calculator is a digital circuit it accepts only binary code in order to process the data. Internally, it uses an encoder circuit to convert decimal number into binary code to understand the machine that which number is pressed. The next number is also converted into its binary code and these two data are processed, suppose addition in this process the machine performs addition by an adder circuit as explained in the last chapter by binary adder. The output of binary adder is a binary code so it cannot be displayed directly. It is converted back to decimal number and displayed by using a display like 7- segment display. The circuit that converts binary code into decimal number is called decoder. In computers it uses keyboard it is accepting ASCII codes its encoder is converting input data into ASCII or BCD to process the computer data. Let us see a simple BCD encoder known as decimal to binary encoder.

BCD Encoder

An encoder is a combinational circuit that converts a standard input into a coded signal. The BCD encoder thus converts a decimal number into BCD code. It uses ten switches representing numbers 0 to 9 just like a calculator keypad. When a particular decimal number is pressed by operating its switch it connects high voltage signal (+5V) to the corresponding OR gates it automatically generates its equivalent BCD code.

For instance, when '0' is pressed refer fig. (2.18) it is kept open from the OR gate circuit the output of encoder will be ABCD = 0000. When switch ' 3' is pressed the C and D OR gates have high inputs; therefore the output of encoder is ABCD = 0011.



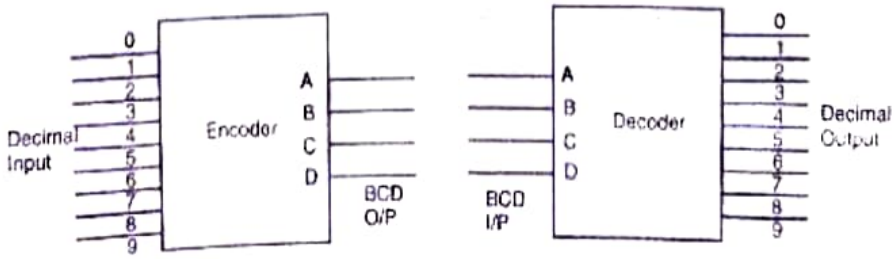
Decimal input	BCD output
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Truth Table

Fig.(2.18) Decimal to BCD Encoder

2.7 DECODERS

It is a combinational logic circuit that converts encoded signal back to its original form. For instance, BCD converts BCD code into decimal number. The decoder circuit performs exactly reverse function; it accepts coded inputs and generates the desired standard output, refer fig. it shows logic symbols of decimal to BCD encoder and BCD to decimal decoder.



BCD to Decimal Decoder

The circuit diagram of BCD to decimal decoder is nothing but the circuit of demultiplexer refer fig. and compare it with BCD to decimal decoder as shown in fig (2.19). The select lines of demultiplexer can be used as BCD inputs DCBA and the data line is removed from the demultiplexer circuit the outputs are many from Y0 to Y9; total ten outputs are there. This circuit is called as 1 of 10 decoder because only one of the ten output lines is high. If the output is active low this can be obtained by replacing AND gates by NAND gates.

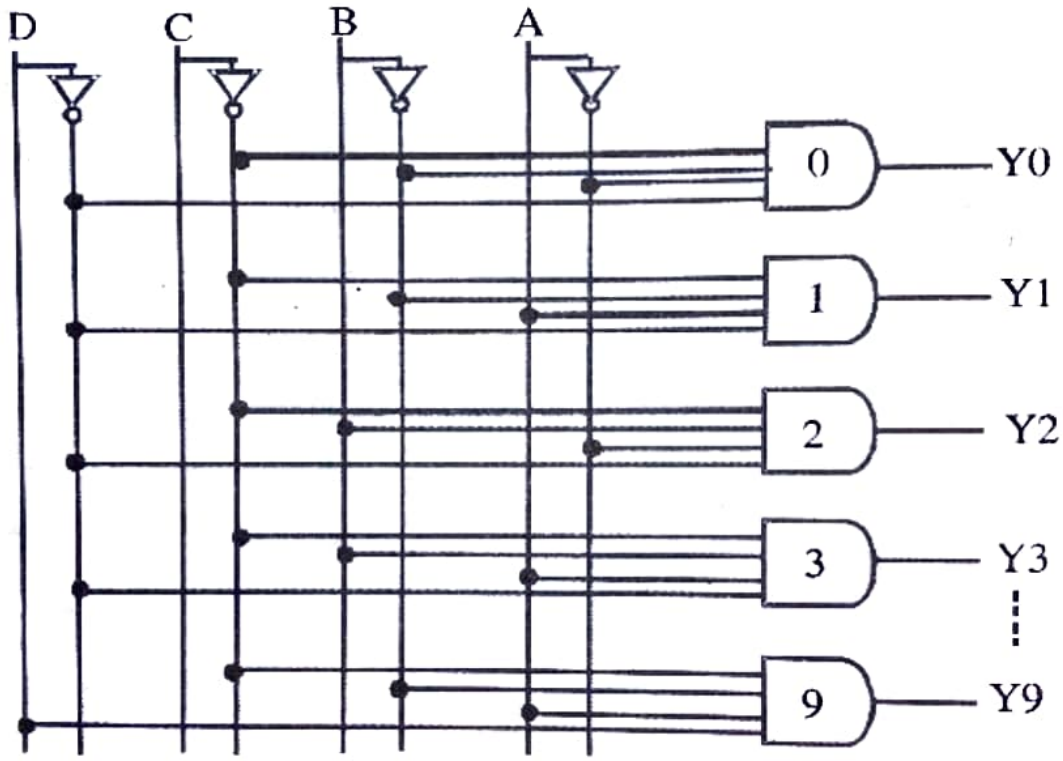


Fig.(2.19)BCD to Decimal Decoder

Working

When BCD input is DCBA = 0000 the first AND gate input is 1111 it enables this AND gate and makes Y0 = 0

i) When BCD input is DCBA = 0001 trace the fig (2.19) the second AND will be enabled and will produce Y1 = 1.

ii) In this way each BCD input produces only one output high when DCBA = 1001 it is equivalent to $[9]_{10}$ now the output Y9 will be high.

THE IC 7446/7447

There are many decoder IC's available the IC7446 is very common BCD to decimal decoder IC basically it is a driver IC because it drives a digital display like LED 7 segment display. IC 7446 hence called as BCD to 7 segment decoder / driver. It has four BCD input pins ABCD refer fig (2.20) and seven outputs (a to g segments). Figure shows IC 7446 is driving a common anode LED display. The truth table for IC

7446/7447 illustrating its driving idea by producing low outputs for a common anode 7-segment display likes FND507.

Truth Table: H = high/logic1 L = low/logic 0

A B C D	a	b	c	d	e	f	g	Display
0 0 0 0	L	L	L	L	L	L	H	
0 0 0 1	H	L	L	H	H	H	H	
0 0 1 0	L	L	H	L	L	H	L	
0 0 1 1	L	L	L	L	H	H	L	
0 1 0 0	H	L	L	H	H	L	L	
0 1 0 1	L	H	L	L	H	L	L	
0 1 1 0	H	H	L	L	L	L	L	
0 1 1 1	L	L	L	H	H	H	H	
1 0 0 0	L	L	L	L	L	L	L	
1 0 0 1	L	L	L	H	H	L	L	

A common anode 7 segment LED display is a matrix display having 7 LED's a,b,c,d,e,f and g. Each LED anode is connected to a common +ve line but the cathode of each LED is driven by IC 7446 as the cathode of each LED is driven by IC7446 as shown in fig (2.20). For instance, when BCD input to IC7446 is ABCD = 0001 it generates low output for the pin 12 (segment b) and pin 11 (segment c) so that these two LEDs glow and displays the number '1' because these two LEDs become forward biased. When

BCD input to IC7446 is ABCD = 0000 it drives the segments a to f except 'g' and displays 0.

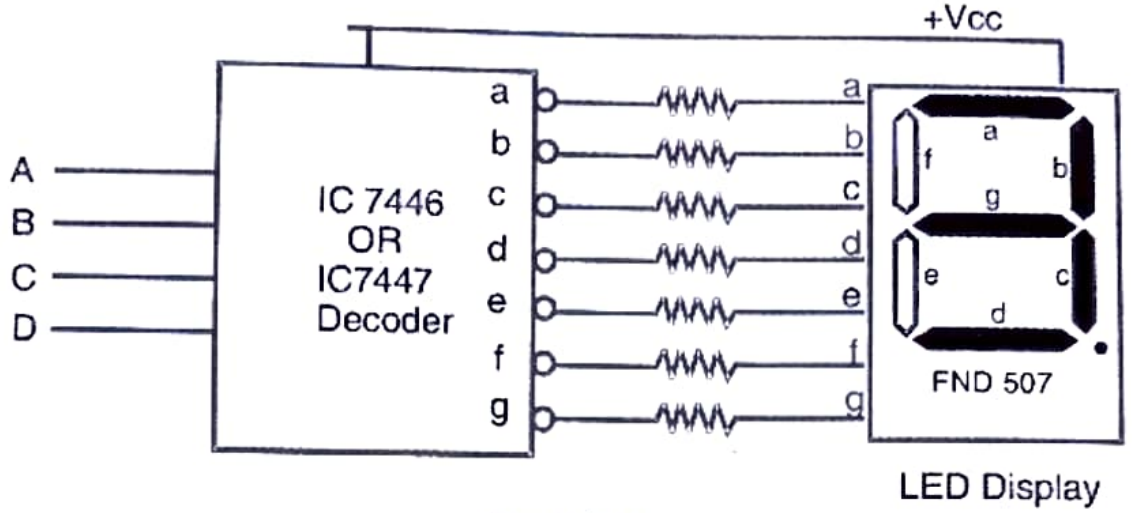


Fig.(2.20)

FLIP-FLOPS AND COUNTERS

A counter is one of the most useful subsystems. Most of the complex digital system contains several counters and register circuits. A number of counters and register circuits can be designed as per requirement. The basic building block of a counter and register is the flip-flop circuit. A counter has input in the form of a chain of pulses. Counters also perform other jobs like dividing the frequency; addressing and can serve as a memory unit. In this topic we shall discuss the various types of flip-flops counters and registers in detail.

Flip-Flop is a bistable multivibrator known as 'latch', where it plays an important role in digital circuits such as registers, counters and memory circuits. As we have studied the gates and their different logic circuits it is very essential to know about different flip-flop circuits and their applications. The flip-flop can be easily constructed by using gates it acts as a memory element which can be operated in '1' state known as "set" or "flip" and the other state can be "0" state known as "reset" or "flop"

Flip-flop has two inputs and two outputs; there are three basic circuits

(1) R-S Flip-flop

(2) D-Flip flop

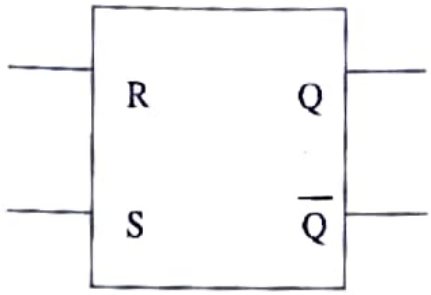
(3) J-K Flip-flop (M/S J-K and T Flip-flop)

(1) R-S Flip-Flop

It is the basic circuit, which requires for constructing the other flip-flop circuits. It has two inputs 'S' and 'R' and two outputs denoted by 'Q' and '$\overline{Q}$' where Q is called normal output and $\overline{Q}$ is called as complementary output.

R	S	Q	$\overline{Q}$
0	0	N/C	N/C
0	1	1	0
1	0	0	1
1	1	Race	

Truth Table

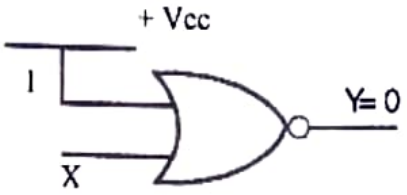


Symbol

2. R-S FLIP-FLOP USING GATES

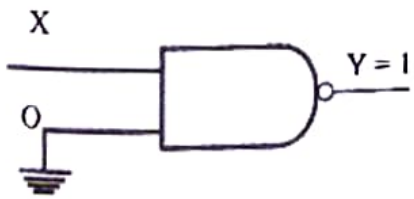
Due to integrated circuit technology the flip-flop can be fabricated in the form of gates. Fig. (5.1) shows a NOR R-S flip-flop with its truth table.

Since NOR and NAND are building blocks, the working of a gate circuit can be explained with their 'disable' and 'enable' state. In gate flip-flop circuit two gates are cross-coupled to latch each other.



Disable state – Refer NOR gate truth table. When one of the inputs is known and if it is high or1, without knowing the other input we can determine its output and it is zero. Hence when one of the inputs to NOR gate is 1 it is 'disabled' its output is zero, irrespective of the other input.

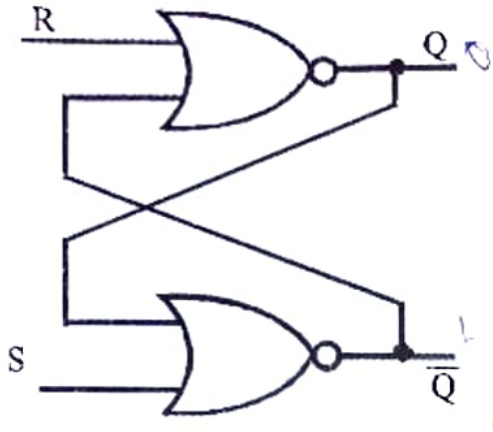
Enable State - When either of inputs of NAND is known and if it is zero its output must be high, it is said that the gate is 'enabled', whatever may be the other input, the output remains high. Refer NAND gate truth table. The gate is enabled when either of inputs of NAND is zero.



Working of NOR Flip-Flop

Case 1: When S = 0 and R = 0, the flip-flop will not change its state, it will remain in the last state hence it is known as "No Change" (N/C) condition.

Case 2: When S = 0 and R = 1 then the upper NOR gate is disabled which will produce Q = 0, now refer the circuit diagram the input to lower NOR gate becomes (Q = 0, S = 0) which will produce $\overline{Q}$ = 1 this condition is known as "Reset". These outputs will latch each other NOR.



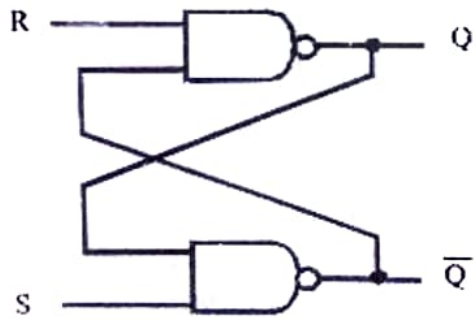
Truth table:

S	R	Q	$\overline{Q}$
0	0	No change	
0	1	0	1
1	0	1	0
1	1	Race/ Forbidden	

Fig. (2.21) R-S Flip-Flop using NOR

Case 3: When S = 1, R = 0 then the lower NOR gate is disabled which will produce $\overline{Q} = 0$, now the input to upper NOR gate becomes ($\overline{Q} = 0$, R = 0) which will produce Q = 1 this condition is knows as "Set" condition and then the circuit is latched.

Case 4: When S = 1 and R = 1 then according to disable state of NOR, both the gates are disabled but practically there is a race between the two NOR gates. In this situation we cannot determine the output state it may be set or reset hence this unpredictable state is known as *"Race"* or forbidden state.



S	R	Q	$\overline{Q}$
0	0	Race/ Forbidden	
0	1	0	1
1	0	1	0
1	1	No Change	

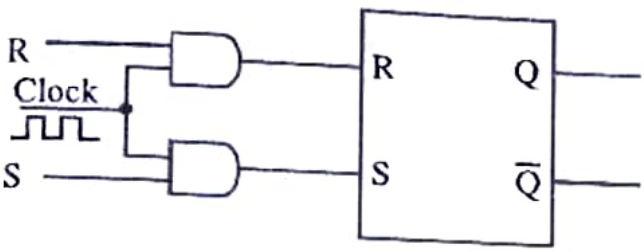
Fig. (2.22) R-S Flip-Flop Using NAND

The working of NAND R-S flip-flop can be explained in similar way, by taking 'Enable' state into consideration. The output for the last state when S = 1 and R = 1, in this case there will be N/C and when S = 0 and R = 0 both NAND gates will try to become Enable that is 'Race' this is the main difference between NOR and NAND R-S flip-flop as shown in the truth table fig. (2.22).

Clocked R-S Flip-Flop

The above circuits are theoretical circuits they transmit inputs to the output without any control. Practically, a control input is connected in addition to (R) and (S) inputs to

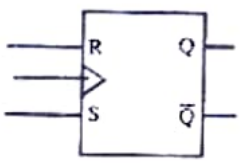
operate the flip flop in synchronisation with the main circuit. The whole system is operated with a common control signal known as "CLOCK". Fig. (2.23) shows a clocked R-S flip-flop and its truth table.



Circuit

CLK	R	S	Q	$\overline{Q}$
0	X	X	N/C	N/C
1	0	0	N/C	N/C
1	0	1	1	0
1	1	0	0	1
1	1	1	Race	

Truth table



Symbol

Fig. (2.23) Clocked R-S Flip Flop (using NOR gates)

Refer the truth table the flip-flop is enabled only when the clock is high. If the clock is low or 0 the flip-flop is inactive the circuit remains in the last state or N/C.
When the CLK is absent or low both AND gates are disabled hence there is no input to the internal R-S flip-flop thus 'No Change' in the output or N/C. When the clock is low the output of the flip-flop remains in its last state irrespective of R and S, it is indicated as X where X is known as *"don't care"* condition.

5.3 CLOCKED D FLIP-FLOP (DELAY FLIP-FLOP)

The D flip-flop can be derived from R-S flip-flop. The R-S flip-flop has two drawbacks:
i) it requires two separate inputs 'S' and 'R'
ii) the possibility of forbidden or race condition.

These drawbacks are eliminated in D flip-flop; this flip-flop needs a single data input 'D'. Fig. (2.24) shows a logic diagram of D flip-flop and its truth table.

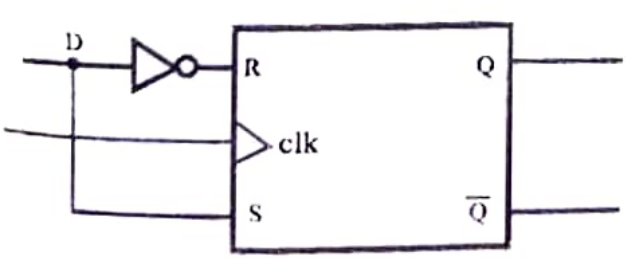
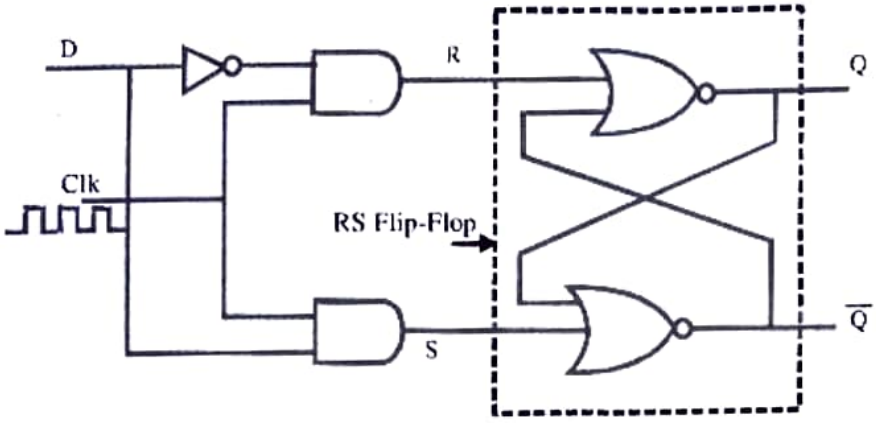


Fig. (2.24) D Flip-Flop

Truth Table:

D	Clk	Q
X	**L**	**Last state**
0	**H**	**0**
1	**H**	**1**

When D = 1 and S = 1 and Q becomes 1.
When D = 0 due to inverter R becomes high, we get low output Q = 0.



CLK	D	Q	$\overline{Q}$
0	X	N/C	N/C
1	0	0	1
1	1	1	0

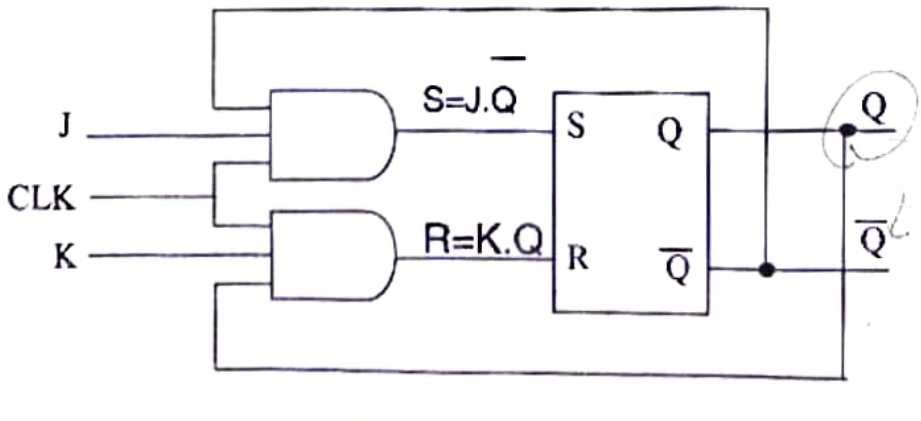
Fig. (2.25) Clocked D Flip-Flop

Fig (2.25) shows a circuit diagram to build D Flip-Flop using NOR R-S flip flop.

When the CLK = 1 and D = 1 the upper AND gate enables the R-S flip-flop, we get Q = 1 and $\overline{Q}$ = 0. When the CLK = 1 and D = 0 it resets the R-S flip-flop. In this way D flip-flop is operated with CLK. The D flip-flops are used in Register circuits ring counters as explained in later topics.

5.4 J-K FLIP-FLOP (JUMP and KEY Flip-Flop)

This flip-flop plays very important role in electronic counter circuits. It also requires R-S flip-flop to build the J-K flip-flop. Consider the last condition when J=1 and K=1 it gives an important output known as *"toggle"* where toggle means, *"Change to opposite state"*. When the J-K flip flop is in toggle state its output changes to the opposite state at every clock input pulse. Fig (2.26) shows its logical circuit diagram and truth table. The circuit diagram does not show the circuit of R-S flip flop.



CLK	J	K	Q	$\overline{Q}$
↓	X	X	N/C	N/C
↑	0	0	N/C	N/C
↑	1	0	1	0
↑	0	1	0	1
↑	1	1	Toggle	

Fig. (2.26) J-K Flip-Flop

When J-K flip-flop is in toggle condition (J = 1 and K = 1) and suppose the last state is Q = 0 and $\overline{Q}$ = 1 then at the first clock (refer circuit diagram) CLK = 1, J = 1 due

to these inputs to the upper AND gate is 111. Its output goes high which will set the flip-flop and now output is Q = 1 and $\overline{Q}$ = 0 which is the opposite state (the lower AND gate is disabled.). Similarly when the next clock arises then the output will change to Q=0 and $\overline{Q}$ =1 in this way the flip-flop is toggled, AND gates are disabled alternately. When J-K flip-flop is used in counter circuits it is operated with edge-triggered clock. It is positive edge or negative edge triggered type. Fig (2.27) shows the symbol of J-K flip flop with other terminals. Edge triggering technique is preferred to avoid false triggering it can be obtained by using a R-C differentiator circuit in series with clock.

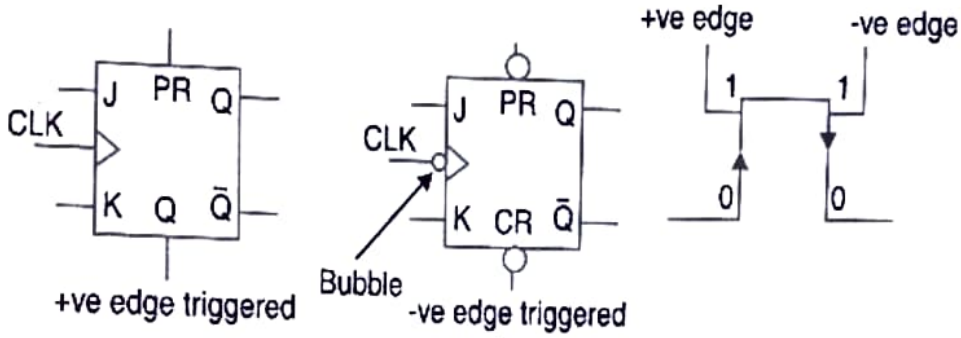


Fig. (2.27)

In the positive edge triggered type flip-flop the output changes at the rising edge of the clock signal or when it changes from 0 to 1. In the negative edge triggered type flip-flop the output changes at the falling edge of the clock signal or when it changes from 1 to 0. The symbol of J-K flip flop also shows two additional inputs known as PR and CR, where PR stands for PRESET and CR for CLEAR. The CLEAR terminal is used to make output low (Q = 0) independent of J, K and the clock. CLEAR is generally used to make counter clear i.e. 0000 like a calculator 'CR' button. The PRESET is used to set the counter to 1111 like down counter.

Circuit with CR and PR facility

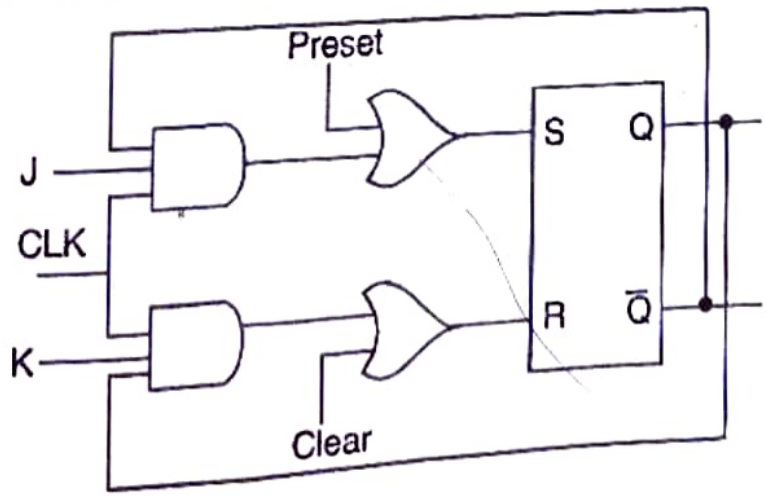


Fig.(2.28)

2.8 COUNTERS

An electronic digital counter is a flip-flop circuit capable of counting the number of clock pulses applied to it. It produces binary output in sequence. The total number of states produced by a counter is denoted by 'M' it specifies modulus of counter. The required number of flip-flops can be calculated by a formula

$$\textbf{Mod of counter} = 2^n \textbf{ or } M = 2^n$$

Where n = no. of flip-flops. For example; for mod-16 counter $2^n = 16$ (n = 4) four flip-flops are required. There are two types counter circuits such as

1) 4-BIT BINARY COUNTER (MOD-16)

One of the most important applications of the flip-flop is counter. Flip-flops can be connected to get an electronic counter. Counter is a circuit, which can be used to count the number of pulses, given to its input. There are a number of counter circuits like, synchronous counter, asynchronous counter, Ring counter, down counter etc. Let us discuss a 4-bit binary counter, which can count, input pulse in form of 4-bit binary number. If we design a counter to count from binary least number 0 0 0 0 to maximum 1 1 1 1, we need a circuit that has 16 different states known as modulus of a counter. For 4-bit binary counter it needs 16 output states, hence it is a (modulus) mod-16 counter. If the number of output states is 3 it is mod-3, if 7 then mod-7 & so on.

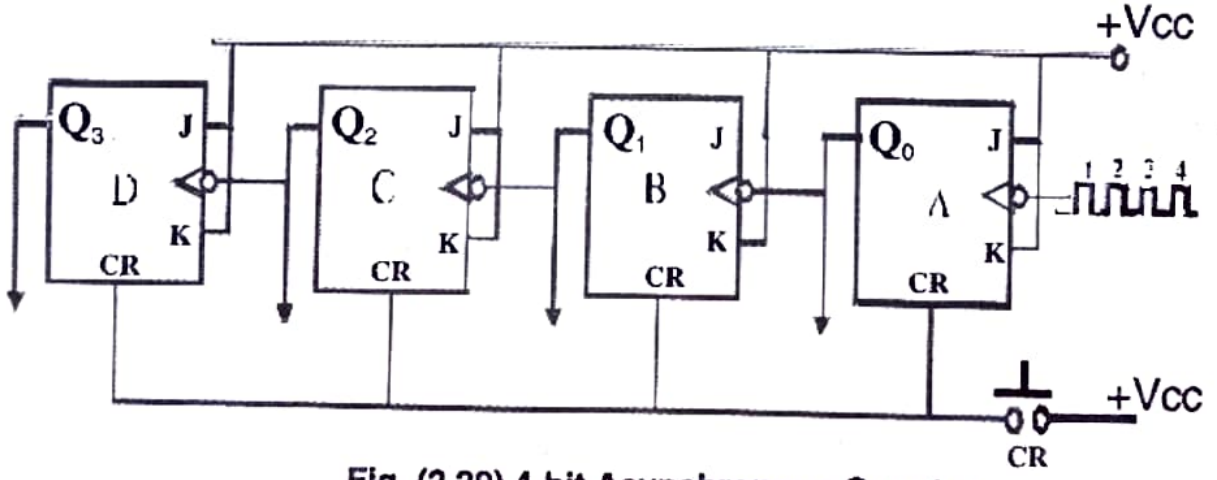


Fig. (2.29) 4-bit Asynchronous Counter

Fig. (2.29) shows a 4-bit binary asynchronous counter. It consists of four JK flip-flops (A, B, C, D,) cascaded and connected in toggle condition (J = K = high). The input is square wave; it is a clock to the flip-flop A. All the flip-flops are negative-edge triggered hence each flip-flop will change its state at negative edge of the clock. Refer fig. (2.29) the symbol of the clock . The outputs Q_3, Q_2, Q_1, Q_0 give binary output. From the flip-flop B to flip-flop D the clock of these flip-flops is nothing but the output of previous flip-flop. For e.g. output Q_1 is the clock to the flip-flop C. The timing diagram of this counter is shown in fig. (2.30). Since the counter is incremented from 0000 to 11111 this is known as *UP counter.*

Working

Assume the outputs are cleared to 0 0 0 0 by using the 'CR' switch, and then the switch is released.

1. As clock pulse 1 arrives at the clock terminal of the flip-flop A, its output goes high or 1 at the negative edge of the first CLK pulse, hence the output Q_0 goes from 0 to 1 which is the positive edge. This transition is the clock to the next flip-flop B, this will remain in the same state. Thus the flip-flop B, C, D, will show the output 000 and the output of the counter Q_3, Q_2, Q_1, Q_0 is 000 1.

2. Now at the second clock pulse see the timing diagram, at the negative edge of this second pulse the flip-flop A will change its state from 1 to 0 which is the negative edge therefore the flip-flop B will change its state from 0 to 1, but it is a positive edge to the next flip-flop and hence no further change. The binary output will be 0 0 1 0. In this way each flip-flop will change its state at the negative edge of its clock pulse. The counter is incremented up-to 1111 and at 17th pulse it returns to its initial state 0000.

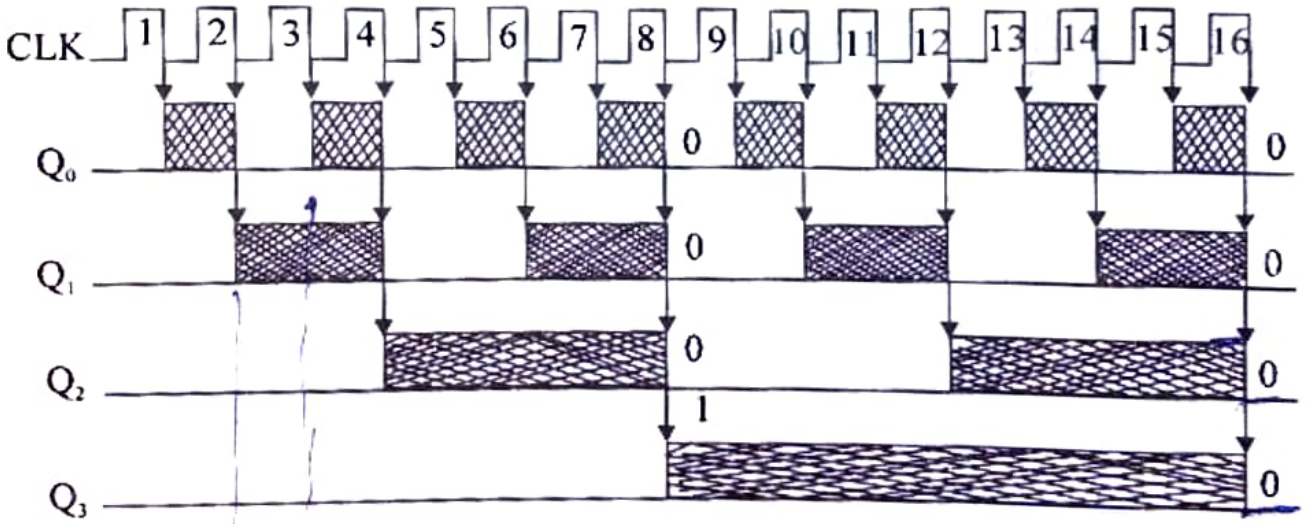


Fig. (2.30) Timing Diagram of 4-bit binary counter

The method to draw the output waveform of each flip-flop is

i) First draw 16 input pulses.
ii) Mark a negative edge of each pulse with an arrow.
iii) Change the state 0 to 1 and 1 to 0 at each negative edge. This will be the output of flip-flop A.
iv) Mark negative edges of this output and draw waveforms for flip-flop B and then for C&D .Now observe the outputs Q3, Q2, Q1, Q0 at any input clock pulse.

In fig. (2.30) pulse no. 8, 16 are illustrated, at pulse number 8 the output is $Q_3Q_2Q_1Q_0$ = 1 0 0 0 which is equivalent of decimal 8 & at 16th pulse each flip-flop goes to 0 state known as counter is in reset condition.
The truth table shows the binary counting which counts input pulses from 0000 to 1 1 1 1 and at 16th pulse it goes to 0 0 0 0 , again at 17th pulse it starts from 0001 and cycle continues.

Truth Table:

Pulse	Q_3	Q_2	Q_1	Q_0	
	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	
9	1	0	0	1	
10	1	0	1	0	
11	1	0	1	1	
12	1	1	0	0	
13	1	1	0	1	
14	1	1	1	0	
15	1	1	1	1	
16	0	0	0	0	Reset
17	0	0	0	1	Recounting

This 4-bit binary counter is called as a ripple and asynchronous counter, which means that all flip-flops do not trigger at the same time. Trigger flows through flip-flops like a ripple on water.
The total number of counts or states through which the counter can progress is given by 2^n where n is the total number of flip-flops. The maximum decimal number of a counter is given by $(2^n) - 1$.

For e.g. for 4-bit binary counter n = 4 flip-flops
$\therefore$ No. of output states $= 2^n = 16$ i.e. 0000 to 1111
And Maximum decimal count $= (2^n) - 1 = 2^4 - 1$
$= 15$ verify this in truth table.

2) DECADE COUNTER (MOD – 10)

The binary counter discussed here is suited for computer use only. However, they are inconvenient to use, because we use decimal numbers from 0 to 9. In a 4-bit binary counter there are 16 counts. To convert this binary counter into a decade counter or decimal counter, suitable modification is done with feedback.

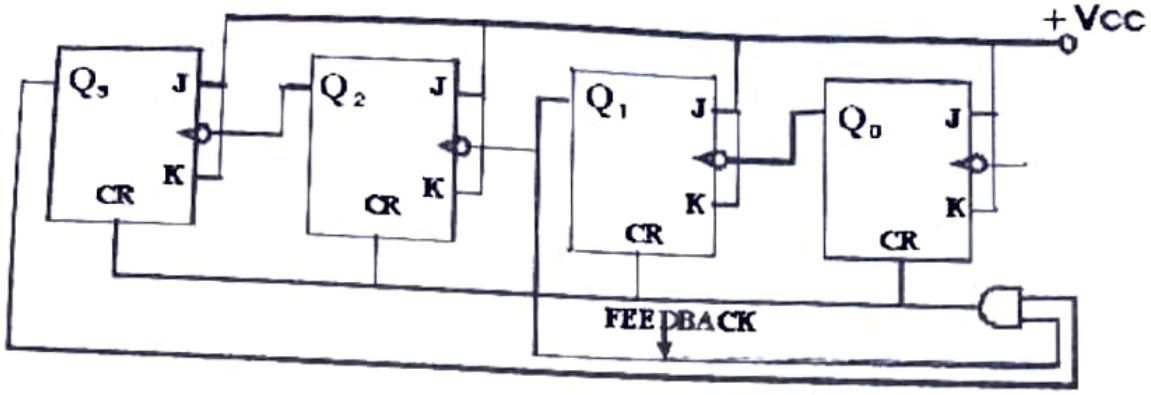


Fig. (2.31) A Modified decade Counter

Decade Counter is the counter that has 10 different output states. In fact, it is a BCD counter, most of the decade counters use 8421 codes, it counts from 0000 to 1001, then it resets to zero. Fig. (2.31) shows a modified decade counter which is 4-bit binary counter with some modification i.e. feedback with gating circuit known as MOD – 10 counter or modulus of 10 counter.

Working

The comparison between 4 – bit binary and decade counter is, the output Q_3 and Q_1 are given to AND gate to get CLR pulse at 10^{th} count. Because at 10^{th} pulse the output would be

Q_3 Q_2 Q_1 Q_0
1 0 1 0

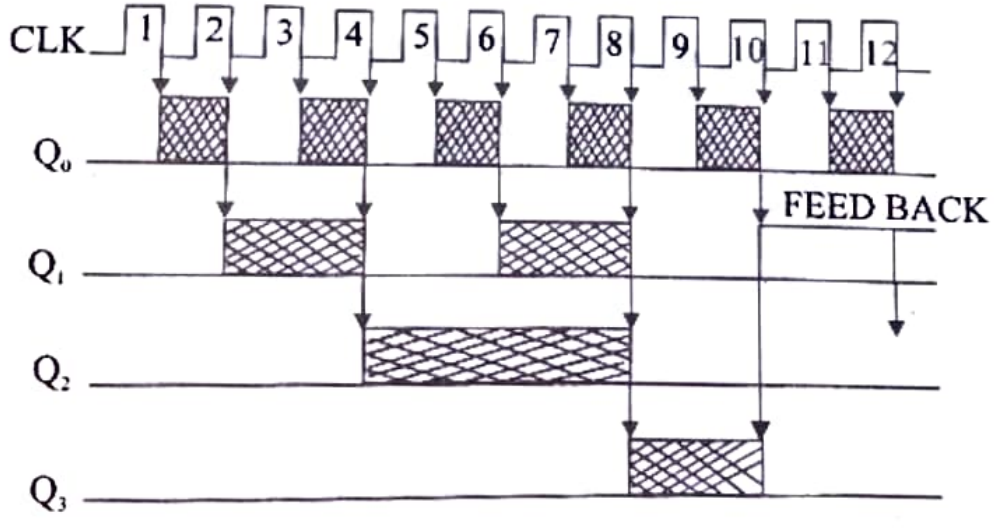


Fig. (2.32) Timing Diagram of Decade Counter

The counter works in normal way as in 4-bit binary counter from pulse 1 to 9 between 0000 to 1001 and observe the timing diagram during these counts Q_3 & Q_1 never become 11 so that AND gate is disabled, it will not give CLR signal to the flip-flops. In other words for the count up to 1001 the AND gate is disabled but at 10^{th} pulse the output of the counter would be 1010 which will make AND gate enable and a high CLR pulse will

reset the flip-flops to 0000. The truth table for decade counter shows the counting and fig. (2.32) shows the timing diagram.

Truth Table

Pulse Count	Q_3	Q_2	Q_1	Q_0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	0	0	→0	0 Reset
11	0	0	→0	1 Restart

This counter may be called as divided by 10 circuits; it gives a 1, high CLR pulse after 10 pulses. Now in industry the standard IC available is '7490' it is a decade counter IC and a standard decoder/driver 7447, which drives the display in response to 7490. If we cascade a counter with decoder and display, a decimal display for 2 digits, 3 digits is possible.

2.9 SHIFT REGISTERS

A Register is a group of flip-flops used to store or rotate the data. The shift register rotates the bits for certain arithmetic operations. It is either shift left or shift right register it uses D flip-flops as explained below.

i) Shift Left Register

Fig.(2.33) shows a shift left register prepared by cascading four D flip-flops. As shown D_{in} sets up the right flip-flop, Q_0 sets up Q_1, Q_1 sets up the third and so on. The flip-flops are +ve edge triggered type so at the +ve edge of the clock the Din move one bit left. refer its timing diagram fig.(2.34a) .

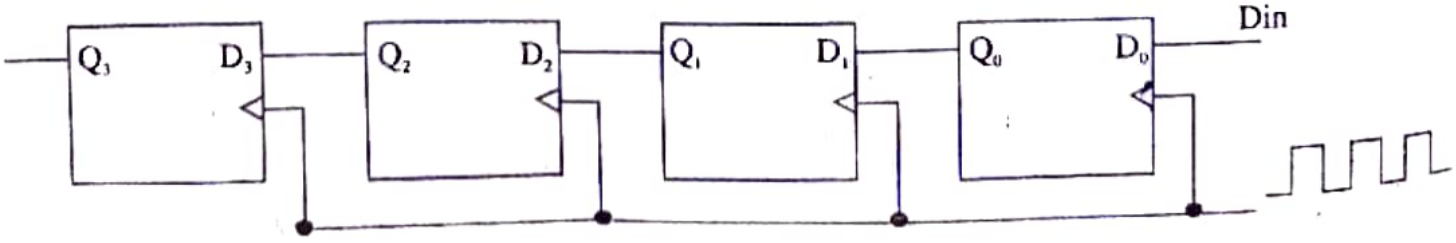
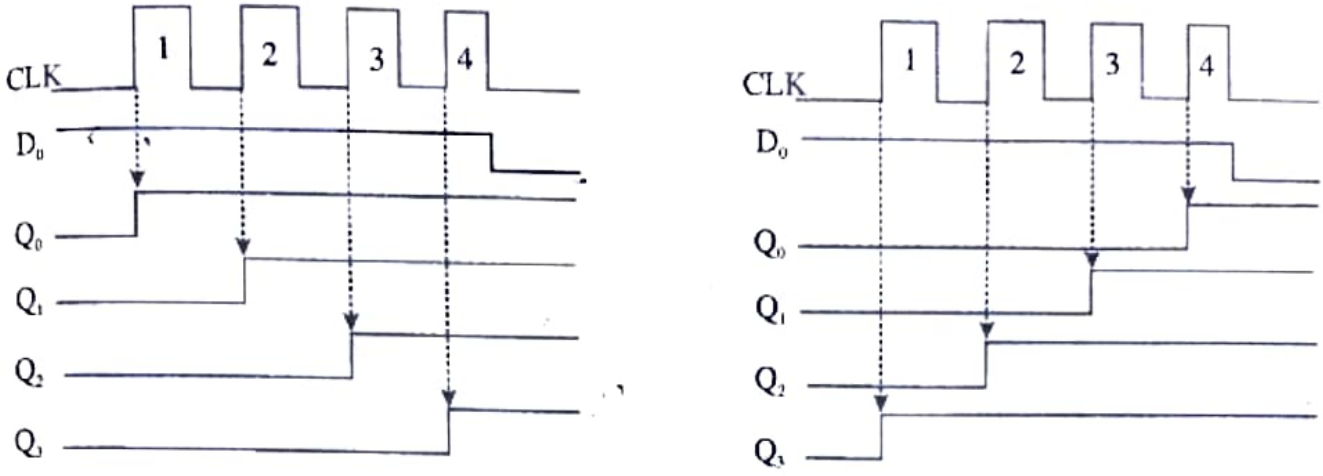


Fig. (2.33) Shift-Left Register

For instance, suppose Din =1 then at the arrival of the first +ve edge of the clock it is shifted to left by the first D flip-flop the other remains unaltered the register output is 0001.At the second clock pulse Q_0 is transferred to its left the output is 0011 by second D flip-flop. The third +ve clock results in 0111 and at fourth 1111.



(a) **Fig.(2.34) Timing diagram of shift registers** (b)

ii) Shift-Right Register

This is an opposite process of shifting the data refer fig.(2.35) this time din is given to D_3, Q_3 to D_2, Q_2 to D_1 and so on. The circuit action is very similar to the shift-left register only the data movement is from left to right as shown in its timing diagram fig.(2.34 b).

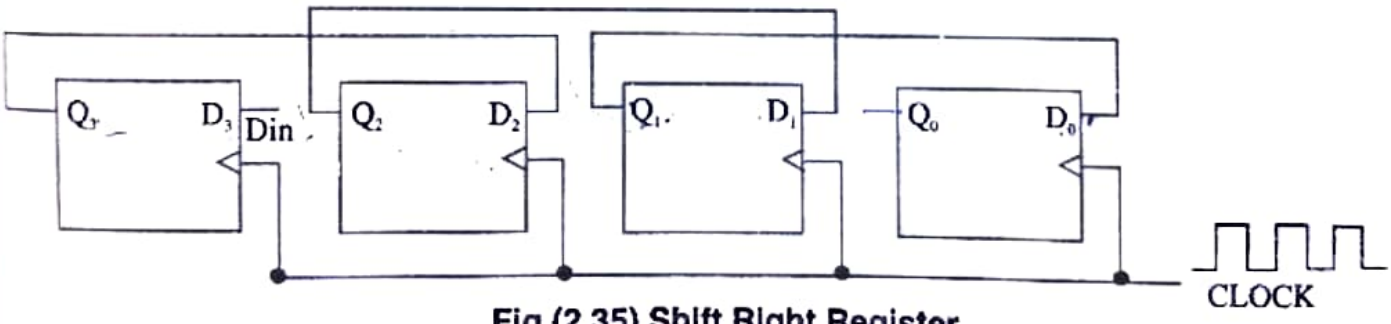


Fig.(2.35) Shift Right Register

QUESTIONS

1. Select the correct answer.

   i) The circuit which has many inputs and one output is ....

   a) Multiplexer b) Demultiplexer c) Encoder

   ii) The 1:16 demultiplexer can be designed with ..... select lines.

   a) 2 b) 4 c) 16

iii) Which of the following circuit can be used as decoder

a) MUX b) DEMUX c) Encoder

iv) In R-S flip flop using NOR if R=1 and S=1 then Q is ....

a) N/C b) Race c) 1

v) 1:32 demultiplexer can be designed with .... select lines.

a) 2, b) 5, c) 16, d) 32

vi) The largest binary number counted by n cascaded flip flops has a decimal equivalent of ...

a) 2^{n-1} b) 2^n c) $2^n - 1$ d) n^2

vii) An electronic decade counter has.....

a) 9 output states b) 11 output states c) 10 output states d) 2 output states.

viii) The maximum binary number counted by 4 bit binary counter is....

a) 0101 b) 1010 c) 1001 d) 1111

ix) Flip-flop is another name for a.....

a) Bistable multivibrator b) Astable multivibrator c) monostable multivibrator

x) In JK flip-flop when J = 1 and K = 1 the output is .....

a) Race b) N/C c) toggled d) high

xi) In JK flip-flop when J = 1, K = 0 and CR = 1 the output is

a) 0 b) 1 c) N/C d) Race

xii) In a priority encoder, if decimal inputs 3 and 6 are activated at the same time, the BCD output corresponds to .....

a) 6 b) 3 c) 6 &3 alternatively

Ans. i) Multiplexer ii) 4 iii) DEMUX iv) Race v) 5 vi) 2^n-1

vii) 10 output states viii) 1111 ix) Bistable Multivibrator x) toggled xi) 0

xii) 6

2. What are the basic requirements of a multiplexer?
3. Draw a neat circuit diagram of 4 to 1 multiplexer and explain how it selects a data line with the help of truth table.
4. What is a demultiplexer? Design a 1:4 Demux using gates and explain its working with truth table.
5. Write the difference between MUX and DEMUX.
6. Draw a neat circuit diagram of Decimal to BCD encoder & explain its working.
7. What is a decoder? Explain one of ten decoder.
9. Give any one application for the following circuits.

a)MUX b) DEMUX c) Encoder d) Decoder

10. What is Flip flop ? State different types with their symbols and truth tables.
11. Design a RS latch using two input NOR gates and explain its working.
12. Draw the circuit of clocked RS flip-flop and explain its working.
13. Explain the working of D flip-flop with truth table and logic diagram.
14. Draw logic diagram of J-K flip-flop and write its truth table.
15. Compare R-S flip-flop with D flip-flop.
16. Draw a neat block diagram of decade counter and explain it with timing diagram.
17. Explain 4-bit Mod-16 binary counter with timing diagram.
18. How may flip-flops are required to construct

i) Mod –3 ii) Mod – 6 iii) Mod – 9 counter

18. What is Register? Explain any one type with timing diagram.
19. With neat diagram, explain how will you use IC7446 as BCD to seven-segment decoder to drive 7-segment display?

❑❑❑

3. FUNCTIONAL HARDWARE OF PC

INTRODUCTION

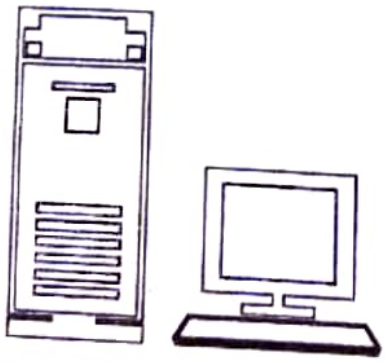
In fact when we talk about a personal computer (PC) and its specifications like Pentium-II, Pentium-III, 500MHz,with modem, 40GB,serial port etc. many people do not know many of these terminologies and it is a difficult job. This chapter explains most of the terminologies and details of hardware components to understand these PC machines, their peripherals and accessories needed to run the machine. After completing this topic one can recognize the hardware of PC and their utilities up to certain extent.

3.1 CONCEPT OF SOFTWARE AND HARDWARE

Computer software is a set of instructions or data anything that can be stored electronically is software. The storage device and display devices are hardware. The distinction between software and hardware is sometimes confusing because they are so integrally linked. Clearly, when you purchase a program, you are buying software. But to buy the software, you need to buy the disk (hardware) on which the software is recorded (loaded). Software is often divided into two categories: **Systems software**: Includes the operating system and all the utilities_that enable the computer to function. **Applications software:** Includes programs that do real work for user. For example, word processors spreadsheet and database management systems fall under the category of applications software.

3.2 PC SPECIFICATIONS

A personal computer (PC) is a type of microcomputer capable of handling single user computer operations becoming very popular in the market. Fig (3.1) shows typical PC system. Most of the PCs are IBM PC. While purchasing a PC user must know few specifications to know about the computer machine and hardware specifications as a general knowledge.



PC with Mini Tower Cabinet

PC with Horizontal Cabinet

Fig. (3.1)

A PC can be used for a number of applications; few of them are Desk-Top-Publishing (DTP), data entry work for various job works, playing video games, programming, drawing etc.

The PC specifications are very important. You can easily get idea of your PC machine such as how much is the faster your PC? What peripherals are available with your PC? And how much is the latest your machine and so on. You must have heard about PC, PC-XT and PC-AT. These are the models; in the beginning there were PCs with very low speed and without hard disks. PC-XT uses the modified and extended version of PC with better speed XT stands for Extended Technology & now we use PC-AT where AT stands for Advance Technology. PC-AT started with 286, 386, 486 then Pentium, Pentium II etc. Now recently, we have Pentium model. All these machines are Intel products the 286, 386 are the microprocessor specifications. Today the most advance and fastest PC is Pentium IV. Normally manufacturer and PC dealer specifies the following technical specifications.

i) Clock frequency specifying the PC speed.
ii) Processor type specifying the Microprocessor type.
iii) Hard Disk capacity
iv) Floppy drive and its size
v) Cache memory (RAM) size.
vi) Extended Memory (RAM) size
vii) Serial and parallel port to connect I/O devices like printer, MODEM, mouse etc.
viii) Modem, multimedia sound blaster card
ix) UPS, TV tuner Card etc.
x) Video monitor type
xi) Printer Type

Let us see two PC types and their specifications.

INTEL I-3 (Latest) PC	Pentium III (Old)PC
Intel I-3 530 (2.93 GHz,4Mb Cache) processor Monitor Type: LED colour Screen Size Up to 19 inches System Memory 2GB HDD capacity Up to 500 GB 8.4 GB Hard Disk USB Keyboard & Mouse SMPS Power Supply 450W	Intel Pentium 550 MHz processor Intel 810 chipset mother board 32 MB RAM, 512 KB cache 15" SVGA Colour Monitor 1.44 MB Floppy Drive 8.4 GB Hard Disk CDROM Drive & Speaker Internal Modem, AGP Card & Sound Card 1KVA UPS (Uninterrupted Power Supply)

3.3 INSIDE PC

Basically the internal hardware of a PC contains the following parts- CPU board (motherboard), hard disk, Power supply, various cards, adapters, connectors and drives. Let us study this internal functional hardware of PC.

Motherboard

A **motherboard** is the physical arrangement in a computer that contains the computer's basic circuitry and components. On the typical motherboard, the circuitry is imprinted or affixed to the surface of a firm planar surface and usually manufactured in a single step. The motherboard is the largest printed circuit board in your computer. It generally houses the CPU chip, the controller circuitry, the bus, and sockets for additional boards, which are called **daughter boards.** The most common motherboard design in desktop computers today is the AT, based on the IBM AT motherboard. A more recent motherboard specification, ATX-AT design. In both the AT and ATX designs, the computer components included in the motherboard are:

i) The microprocessor
ii) (Optionally) coprocessors
iii) Memory
iv) Basic input/output system
v) Expansion slot
vi) Interconnecting circuitry

Additional components can be added to a motherboard through its expansion slot. The electronic interface between the motherboard and the smaller boards or cards in the expansion slots is called the bus. In a computer, a bus is the data path on the computer's motherboard that interconnects the microprocessor with attachments to the motherboard in expansion slots (such as hard disk drives, CD-ROM drives, and graphics adapters).

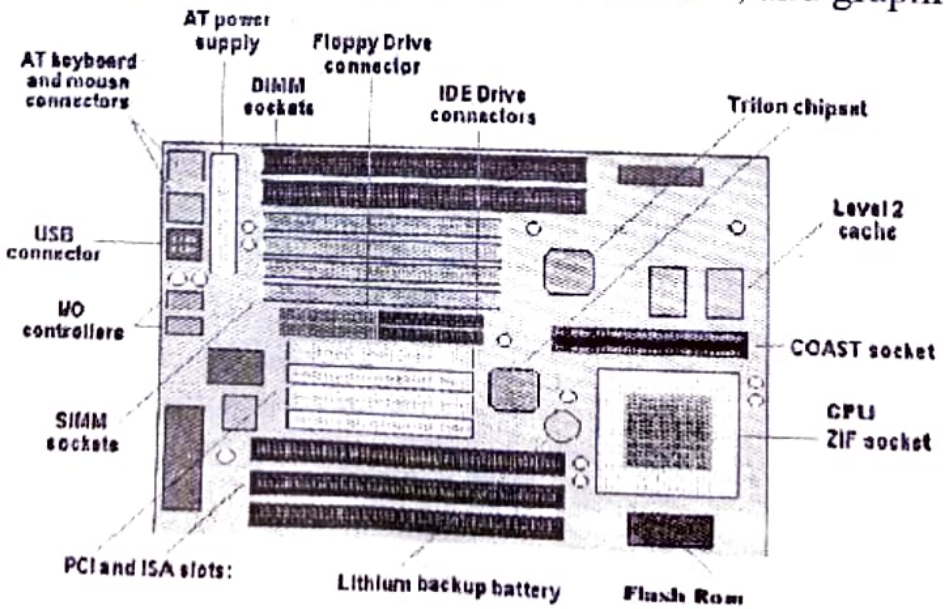


Fig. (3.2) A Typical Motherboard Layout

3.4 CPU PROPERTIES

The performance of a PC depends on the CPU properties. In fact it is decided by Microprocessor specifications. Today most of the PCs use Intel's Microprocessor 8086 family like 80386,80486 Pentium etc. The microprocessor/CPU properties are expressed in a standard format such as follows. The important properties are

i) **Clock Frequency (CPU Speed):** It specifies the execution speed of CPU it is operated with a clock. The typical clock frequency range of Pentium is 500 MHz to 1GHz while that of 80486 is 66MHz. It means Pentium speed is much more than 80486 resulting in excellent performance.

ii) **Data Bus width:** It specifies the word size in terms of number of lines available for carrying data at a time. It is 8-bit, 16 bit or 32 bit. Pentium has widest data bus it is 32-bit and 16-bit for 80386. Pentium has maximum efficient word processing capacity.

iii) **Address Bus:** It specifies the memory accessing capacity. The CPU can access a memory by putting address on address bus. The address bus for Pentium is 32-bit able to access 2^{32} =1Gb memory while 8085 has only 16-bit address bus. 8085 microprocessor can access only 2^{16} = 64kb memory.

iv) **Serial port:** It specifies whether serial communication is provided or not? Because peripheral may require serial data transfer.

3.5 TYPES OF MEMORIES

Basically Memory is a vast topic; it is very difficult to classify them. PC is accessing memory for data and program transactions. Depending on the volume, data type, and many other factors CPU has to use various types of memories. This topic explains these memories with their features and limitations.

Conventional memory

Conventional memory is the first 640Kb of memory. DOS based programs load and run from within the first 640Kb of memory. On DOS systems, conventional memory refers to the portion of memory that is available to standard DOS programs. DOS systems have an address space of 1MB (megabyte), but the top 384K.

A certain amount of the random access memory was reserved as a place to load in the basic input/output system or initializing program, another space was reserved for buffer areas for display data, and another space for interrupt data, and so forth. The remaining 640 kilobytes of memory could be used by the DOS operating system(which was relatively small), and application programs.

Expanded Memory

Also known as EMS (Expanded Memory Specification), expanded memory is a technique for utilizing more than 1MB (megabyte) of main memory in DOS -based computers. This generally must be addressed by software using the Expanded Memory Specification (EMS). Hence, the need for an EMS driver or some sort of software is to handle this task. The limit of 1MB is built into the DOS operating system. The upper 384K is reserved for special purposes, leaving just 640K of conventional memory for programs. There are several versions of EMS. The original versions, called EMS 3.0 and 3.2, enable programs to use an additional 8MB of memory, but for data only. An improved version developed by AST is known as EEMS (Extended EMS). EEMS enables programs to use extra memory for code as well as for data. The most recent version of EMS is known as EMS 4.0 or LIM 4.0, LIM being the initials of the three companies that developed the specification: Lotus, Intel, and Microsoft. EMS 4.0 raises the available amount of memory to 32MB.

Extended Memory

Until the release of Microsoft Windows 3.0 in 1990, expanded memory was the preferred way to add memory to a PC. The alternative method, called extended memory, was less flexible and could be used only by special programs such as RAM disks. Windows 3.0 and all later versions of Windows, however, contain an extended memory manager that enables programs to use extended memory without interfering with one another. In addition, Windows can simulate expanded memory for those programs that need it (by using the EMM386.EXE driver).

In modern systems, the memory that is above 1 MB is used as extended memory (XMS).

Extended memory is the most "natural" way to use memory beyond the first megabyte, because it can be addressed directly and efficiently.

Physical Vs Logical memory

Physical refers to anything related with hardware. The opposite of physical is logical or virtual, which describe software objects. For example, physical memory refers to the actual RAM chips installed in a computer. Virtual memory, on the other hand, is an imaginary storage area used by programs.

e.g. A physical data structure refers to the actual organization of data on a storage device. The logical data structure refers to how the information appears to a program or user. For example, a data file is a collection of information stored together. This is its logical structure. Physically, however, a file could be stored on a disk in several scattered pieces. Same concept is applied for memory as well.

Virtual memory

An imaginary memory area supported by some operating systems (for example. Windows but not DOS) in conjunction with the hardware. You can think of virtual memory as an alternate set of memory addresses. Programs use these virtual addresses rather than real addresses to store instructions and data. The address space represents the amount of memory available to a program. When the program is actually executed, the virtual addresses are converted into real memory addresses. The purpose of virtual memory is to enlarge the address space, the set of addresses a program can utilize. For example, virtual memory might contain twice as many addresses as main memory.

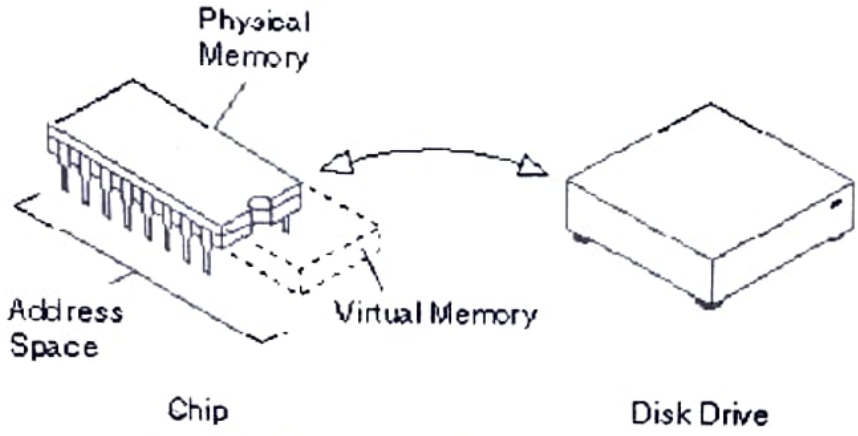


Fig.(3.3) Concept of Virtual Memory

A program using all of virtual memory, therefore, would not be able to fit in main memory all at once. Nevertheless, the computer could execute such a program by copying into main memory those portions of the program needed at any given point during execution. To facilitate copying virtual memory into real memory, the operating system divides virtual memory into pages, each of which contains a fixed number of addresses. Each page is stored on a disk until it is needed. When the page is needed, the operating system copies it from disk to main memory, translating the virtual addresses into real addresses. The process of translating virtual addresses into real addresses is called mapping. The copying of virtual pages from disk to main memory is known as paging or swapping.

3.6 SEMICONDUCTOR MEMORIES

Semiconductor memory chips are available in various configuration their capacity is specified in terms of bytes e.g. 1K, 32 K byte, 64 K byte, 1 M byte, 1Gbyte etc. Where 1 K memory chip is a memory that has 1024 cells.

These Memories can generally be divided into two main groups and further they are classified into other subgroups.

A) RAM (Random Access Memories)
   i) Static RAM ii) Dynamic RAM

B) ROM (Read Only Memories)
   i) Mask ROM ii) PROM iii) EPROM iv) EAPROM.

RAM

RAM (Random Access Memory) is the place in a computer where the operating system, application programs, and data in current use are kept so that they can be quickly reached by the computer's processor. It is used for temporary storage. When switched OFF, any information stored in RAM is lost. These memories are also known as Read/Write memories or scratch pad. When it is required to write the binary data in to the memory or to read the data from the memory of any random location, similarly when it is required to change the data frequently then this type of memory is preferred.

Each RAM cell is a simple bipolar or MOS flip-flop. Since the loss of power means loss of data, when supply becomes OFF whatever data is stored in RAM is lost, it is known as **"volatile"** memory.

RAM is much faster to read from and write to than the other kinds of storage in a computer, the hard disk, floppy disk, and CD-ROM. However, the data in RAM stays there only as long as your computer is running. When you turn the computer OFF, RAM loses its data. When you turn your computer on again, your operating system and other files are once again loaded into RAM, usually from your hard disk.

RAM can be compared to a person's short-term memory and the hard disk to the long-term memory. If RAM fills up, the processor needs to continually go to the hard disk to overlay old data in RAM with new, slowing down the computer's operation. Unlike the hard disk, which can become completely full of data so that it won't accept any more, RAM never runs out of memory. It keeps operating, but much more slowly than you may want it to. RAM is available in to types Static and Dynamic. The dynamic RAM is more powerful and big storage capacity consumes small power than static RAM.

ROM

Read Only Memory is "built-in" computer memory containing data that normally can only be read, not written to. ROM contains the programming that allows your computer to be "booted up" or regenerated each time you turn it on. A small long-life battery in your computer sustains the ROM. If you ever do the hardware setup procedure with your computer, you effectively will be writing to ROM.

ROM (Read Only Memory) would play an important role in making computers easier to use. Unlike RAM (Random access memory) ROM would not lose it's contents when the power was turned OFF hence called as **Non-Volatile**. Early microcomputers lacked ROM memory and had to be started by entering a bootstrap program, or sometimes the whole program that the user would want to run, directly into the computers memory via switches on the front panel. ROMs enabled computers to "auto start".

One of the most important programs stored in the ROM is the BIOS (Basic Input Output System). It is the program letting your application programs and the hardware interacting with each other.

The BIOS also contains a program called Power-On-Self-Test. This checks the motherboard and other devices connected to computer during the system power on time. In ROM the manufacturer stores information. Once it is stored, the memory cannot be erased or changed. The computer can only read this memory to carryout the various operations.

PROM

Programmable read-only memory (PROM) is read-only memory (ROM) that can be modified once by a user. The basic difference between ROM and PROM is, in case of ROM manufacturer decides the data of the memory and in case of PROM user can decide the data of memory by special method of programming. But once PROM is programmed it cannot be altered again. It can be programmed only once. This is also a nonvolatile all types of ROMS are nonvolatile. Special PROM programs are available but these are more expensive.

PROM is a way of allowing a user to tailor a micro code program using a special machine called a PROM programmer. This machine supplies an electrical current to specific cells in the ROM that effectively blows a fuse in them. The process is known as burning the PROM. Since this process leaves no margin for error, most ROM chips designed to be modified by users use erasable programmable read-only memory (EPROM) or electrically erasable programmable read-only memory (EEPROM).

EPROM

The main feature of these memories is re-programming and erasing of data is possible. Hence name is erasable PROMS. EPROM can be programmed number of times. The user can erase the data by exposing them to ultraviolet light. But the main difficulty in the process of erasing is the user must disconnect the memory from the main circuit. Another difficulty while erasing the old data if it is not erased properly then the user cannot correct or erase the old data after programming the new data. Also the time required for reprogramming is comparatively more.

EPROM (Erasable Programmable Read-Only Memory) is programmable read-only memory (programmable ROM) that can be erased and re-used. Erasing is caused by applying an intense ultraviolet light through a window that is designed into the memory chip. Although ordinary room lighting does not contain enough ultraviolet light to cause erasure, bright sunlight can cause erasure. For this reason, the window is usually covered with a label when not installed in the computer.

A different approach to a modifiable ROM is electrically erasable programmable read-only memory (EEPROM).

EEPROM

EEPROM (electrically erasable programmable read-only memory) is user-modifiable read-only memory (ROM) that can be erased and reprogrammed (written to) repeatedly through the application of higher than normal electrical voltage. Unlike EPROM chips, EEPROMs (E^2PROM) does not need to be removed from the computer to be modified. However, an EEPROM chip has to be erased and reprogrammed in its entirety, not selectively. It also has a limited life - that is, the number of times it can be reprogrammed is limited to tens or hundreds of thousands of times. In an EEPROM that is frequently reprogrammed while the computer is in use, the life of the EEPROM can be an important design consideration.

A special form of EEPROM is flash memory, which uses normal PC voltages for erasure and reprogramming.

3.7 CONTROLLER

A device that controls the transfer of data from a computer to a peripheral device and vice versa. For example, disk drives, display screens, keyboards, and printers all require controllers.

In personal computers, the controllers are often single chips. When you purchase a computer, it comes with all the necessary controllers for standard components, such as the display screen, keyboard, and disk drives. If you attach additional devices, however, you may need to insert new controllers that come on expansion boards.

Controllers must be designed to communicate with the computer's expansion bus. There are three standard bus architectures for PCs -- the AT bus, PCI (Peripheral Component Interconnect), and SCSI. When you purchase a controller, therefore, you must ensure that it conforms to the bus architecture that your computer uses.

Disk controller

A chip and associated circuitry that is responsible for controlling a disk drive. There are different controllers for different interfaces. For example, an IDE interface requires an IDE controller and a SCSI interface requires a SCSI controller. On Macintosh computer systems, the disk controller is built into the motherboard.

A controller for a hard disk is called a hard disk controller (HDC), and a controller for a floppy disk is called a floppy disk controller (FDC).

Direct Memory Access (DMA)

When a large amount of data is to be transmitted between the processor and any external device, routing them through the processor is a very slow process.e.g. if it is transfer of data from memory to the disk drive, reading from memory and writing to the

external device byte by byte is a very slow process which slows down the fast I/O devices.

So direct memory access is a method used by computer designers to overcome above problem, for this special device called DMA Controller is placed in between processor and I/O device. So this device takes the control of system bus and bypasses processor completely and does data transfer between I/O devices.

3.8 TYPES OF SYSTEM BUSES

The components inside your computer talk to each other in various different ways. Most of the internal system components, including the processor, cache, memory, expansion cards and storage devices, talk to each other over one or more "buses".

A bus, in computer terms, is simply a channel over which information flows between two or more devices (technically, a bus with only two devices on it is considered by some a "port" instead of a bus). A bus normally has access points, or places into which a device can tap to become part of the bus and devices on the bus can send to and receive information from other devices.

Data and Address Buses

Every bus is composed of two distinct parts: the *data bus* and the *address bus*. The data bus is what most people refer to when talking about a bus; these are the lines that actually carry the data being transferred. It is a **bi-directional** bus.

The address bus is the set of lines that carry information about where in memory the data is to be transferred or from. It is a **uni-directional** bus CPU sends address on this bus.

In addition, there are a number of control lines that will control how the bus functions, and allow users of the bus to signal when data is available.

Control Bus:

The control bus is used to send different control signals like read write signals to the memory, signals for serial data transmission etc. It can produce some synchronization signals as well. Some control signals sent on this control bus are used to identify the type of device, processor wants to interact with.

Universal Serial Bus

A two-way, high-speed serial interface to which you can dynamically connect and disconnect appliances that are automatically recognised by the computer

Universal Serial Bus (USB) has been around for a few years now and USB ports are fitted to just about every computer now. With the help of it, up to 127 devices can be connected on one port.

ISA Bus:

Industry Standard Architecture. It allows 16 bits at a time to flow between the motherboard circuitry and an expansion slot card and its associated device(s). ISA runs at 8 MHz. The ISA bus is a serious bottleneck. It is used for slow speed applications like modems and sound cards

EISA Bus

EISA is a standard (computer interconnection) architecture that extends the ISA to a 32-bit interface. It was developed in part as an open alternative to the proprietary Micro Channel Architecture (MCA). EISA data transfer can reach a peak of 33 megabytes per second.

Micro Channel Architecture (MCA) Bus

The MCA bus (also called the *Micro Channel* bus; MCA stands for "Micro Channel Architecture") was IBM's attempt to replace the ISA bus with something "bigger and better". When the 80386DX was introduced in the mid-80s with its 32-bit data bus, IBM decided to create a bus to match this width. MCA is 32 bits wide, and offers several significant improvements over ISA. (One of MCA's disadvantages was rather poor DMA controller circuitry). MCA automatically configured adapter cards, so there was no need to fiddle with jumpers

VESA Local Bus (VLB)

The first local bus to gain popularity, the *VESA local bus* (also called *VL-Bus* or *VLB* for short) was introduced in 1992. VESA stands for the Video Electronics Standards Association, a standards group that was formed in the late eighties to address video-related issues in personal computers. Indeed, the major reason for the development of VLB was to improve video performance in PCs.

The VLB is a 32-bit bus, which is in a way a direct extension of the 486-processor/memory bus. A VLB slot is a 16-bit ISA slot with third and fourth slot connectors added on the end. The VLB normally runs at 33 MHz, although higher speeds are possible on some systems. Since it is an extension of the ISA bus, an ISA card can be used in VLB slots of a VLB video card and I/O controller greatly increases system performance over an ISA-only system. Today VLB is obsolete for new systems

PCI Local Bus

Peripheral Connection Interface through this slot you make a connection to the PCI bus, which is "Local" to the CPU. The PCI bus transfers data at higher speeds to and from the devices. PCI is 32 bits wide and runs at 33 MHz. Used for most modern expansion cards, like high-speed monitors, 3D graphics cards, modems etc. It is geared specifically to fifth- and sixth-generation systems, although the latest generation 486

motherboards use PCI as well. The key to PCI's advantages over its predecessor, the VESA local bus, lies in the chipset that controls it. The PCI bus is controlled by special circuitry in the chipset that is designed to handle it.

3.8 INTRODUCTION TO INTERRUPTS

A signal informing at that an event has occurred. When a program receives an interrupt signal, it takes a specified action (which can be to ignore the signal). Interrupt signals can cause a program to suspend itself temporarily to service the interrupt. Interrupt signals can come from a variety of sources. For example, every keystroke generates an interrupt signal. Interrupts can also be generated by other device such as a printer to indicate that some event has occurred. These are called hardware *interrupts*. Interrupt signals initiated by programs are called software *interrupts*. A software interrupt is also called a *trap* or an *exception*. PCs support 256 types of software interrupts and 15 hardware interrupts. Each type of software interrupt is associated with an *interrupt handler* -- a routine that takes control when the interrupt occurs. Routine is a section of a program that performs a particular task. Programs consist of module each of which contains one or more routines. The term *routine* is synonymous with procedure function, and *subroutine*). For example, when you press a key on your keyboard, this triggers a specific interrupt handler. The complete list of interrupts and associated interrupt handlers is stored in a table called the interrupt vector table which resides in the first 1 K of addressable memory.

QUESTIONS

1. Select the correct alternatives.

a) The main system board in a PC is known as.....
   i) CPU ii) Motherboard iii) Controller board

b) Conventional memory is..... in size.
   i) 640 Kb ii) 1 Mb iii) 32Kb

c) An imaginary memory is known as..... memory
   i) Expanded ii) virtual iii) conventional

d) ..... is a Volatile memory
   i) RAM ii) PROM iii) ROM

e) ..... bus is unidirectional
   i) Data ii) Address iii) none of these

f) MCA bus is ....wide
   i) 32-bit ii) 16-bit iii) none of these

g) .....as a non-erasable memory
   i) RAM ii) PROM iii) EEPROM

h) DMA allows ..... data transfer
   i) Fastest ii)slowest iii) none of these

j) An interrupt generated by an external device like printer is known as....interrupt
   i) Software ii) hardware iii) trap.

Answer: a) Motherboard b) 640 Kb c) virtual d) RAM e) Address f) 32-bit
g) PROM h) Fastest i) hardware j) hardware

2. State True or False:

i) PC-AT stands for PC with Advance Technology.

ii) Pentium has 16-bit data bus.

iii) The DOS based programs are always stored in conventional memory.

iv) ROM is a Volatile memory.

v) DMA increases the rate of data transfer.

Answer: i) True ii) False iii)True iv)False v) True.

3. Write the difference between Software and Hardware.
4. Enlist the components and slots, which are normally assembled on a motherboard.
5. What do you mean by conventional memory?
6. Write the difference between physical and logical memory
7. State different types of semiconductor memories and explain any two.
8. Explain the difference between volatile and non-volatile memories. Give one example.
9. Explain different types of non-volatile memories.
10. What do you mean by controller? Give suitable example.
11. State different types of system bus and explain any two.
12. Explain the concept of Interrupt.

❑❑❑

4. PERIPHERAL DEVICES

INTRODUCTION

Computer system mainly designed to accept input data through input device process it and to display or to print the results. It may be required to store this result on a memory. To get efficient and fast operation, CPU requires suitable input, output and storage devices. This chapter explains most popular and commonly used peripheral devices with illustrative diagrams. Since they are connected to CPU are called auxiliary or peripheral devices.

4.1 INPUT DEVICES

Several different devices can be used as input devices such as keyboard, floppy disk, magnetic disk, punched paper tape, etc. There are some devices which are used for both input as well as output such as floppy disk they are called Input-Output or I/O devices. Let us classify them according to their applications

1. Keyboard (Input device)
2. Punched card (I/O device)
3. Magnetic Floppy Disk (I/O device)
4. Mouse (Input device)
5. Scanner (Input device)
6. VDU (Output device)
7. Printer (Output device)

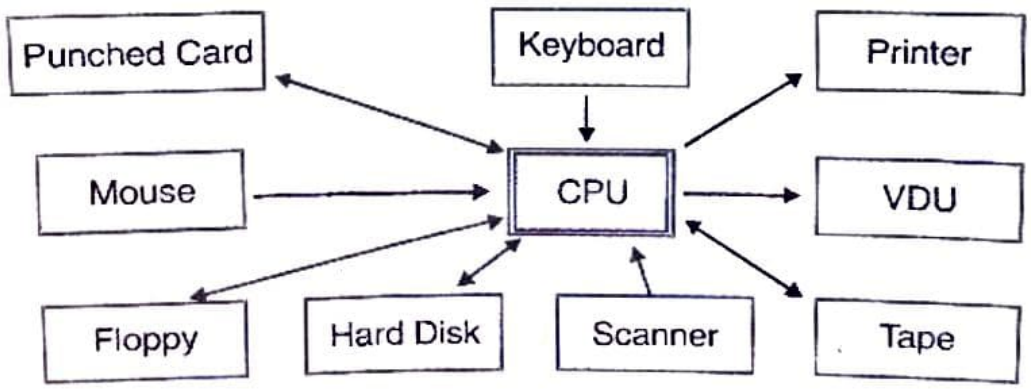


Fig. (4.1) Peripheral I/O devices

a) KEYBOARD

It is a most common online input device like typewriter. It allows all alphanumeric set 0 to 9, A to Z in small and capital form with a few control keys like Shift, Spacebar, Enter, etc. Keyboard is available in two main types.

1) 84 keys 2) 101 keys enhanced type

The enhanced type is more popular keyboard with PCs. It contains additional key than a typewriter such as

a)Arrows (Home, PgUp, PgDn, End) b)Function keys (F1 to F12)
c)Editing keys (Backspace, Delete, Insert) d)Modifier keys (Shift, Alter, Control)
e)Enter f)Other special keys

It converts each key enclosure into a ASCII code and then it is stored in the form of binary. To connect keyboard an interface is required for matching and conversion purpose. There are following IBM type keyboard interfaces.

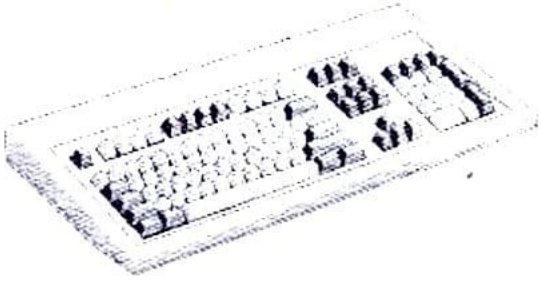
1) Original PC or XT keyboard interface
2) Original AT keyboard interface
3) The 101 keys PS/2 keyboard, generally called enhanced keyboard.

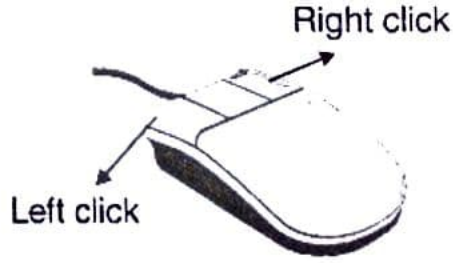
Switch / Contact Design

According to mechanism of switch contact design there are basic two kinds of keyboards, a) Switch based, b) Capacitive type.

In switch-based keyboards, there is micro switch for each key on the keyboard. Switches can become dirty but its replacement is easy.

The IBM and AT & T keyboards are capacitive. In the bottom of the keyboard there is one large capacitive surface. Each keyboard key pushes a spring, which in turn pushes paddle. The paddle makes an impression on capacitive surface. This surface sends a signal, which is interpreted by 8048 microprocessesor on keyboard. It sends scan-code to PC and PC interprets that key.





Keyboard **Fig. (4.2)** **Mouse**

b) MOUSE

Mouse is a pointing input device. It has at least two or three buttons. A mouse is a rectangular plastic box with rubber ball at its center from the bottom side. It looks like a mouse as shown in fig,(4.2). This device is used in conjunction with a video monitor and operated by two mutually perpendicular wheels, which rests on a flat surface. Two wheels are monitored electronically. The two wheels are perpendicular to each other, so one tracks X-axis motion and one tracks Y-axis motion. The third wheel balances the first two. Every mouse is having ball at the bottom of mouse. It is also having three wheels, which turn when ball moves against them. It allows operating computer operations without giving instructions but it requires menu driven software like Windows. By moving the cursor one can select the required menu and by click can operate the menu.

Generally actions related to mouse are as follows,

1) **Single click**

Press left mouse button or right mouse button. It is used for generally selecting any option.

2) **Double click**

Clicking two times is double clicking. It is generally for opening any menu.

3) **To Move**

By moving the mouse itself the mouse-pointer or curser position can be positioned anywhere.

4) **Dragging**

By selecting a window we can change the position of Window known as dragging. While dragging select the Window by pressing the left key and after changing the position release the key. Generally this is used for moving window or graphical operations.

c) SCANNER

Scanner is a small and economic machine like Xerox but it converts documents, photographs, images in to computer data and the data can be stored in to PC for editing or printing for desired applications. This is available in colour mode also. Scanner quality depends on DPI (dot/inch) its resolution is much better because its DPI rating is 1200 and above.

It is available in three models

i) Hand held ii) Flat bed Scanner

iii) Paper-Feed Scanner

Scanner is an input device it uses optical character reader to and detects presence of ink. It compares it with standard character and copies on computer storage. In case of Photographs it scans dot by dot and stores it as an editable image.

4.2 OUTPUT DEVICES

Most essential and live displaying output device is Video, which is the part of computer system. A printer is also important output peripherals device usually required to print results, documents and processed data. Let us study these output devices with their features.

a) VIDEO (MONITOR)

The video i.e. display on a computer is the primary output device. The primary objective of video board is to take information from CPU and display it on a monitor.

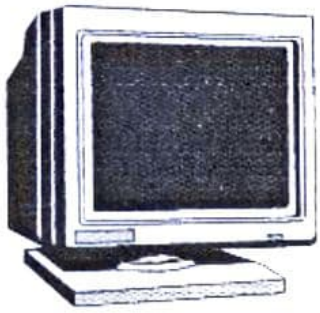
Before 1987 video standard basically come from IBM but after 1987 PS/2 was successful. The top video in 1987 was the VGA, which is a base monitor today. There are following kinds of older monitors.

1) Composite monochrome 2)Composite colour 3)RGB

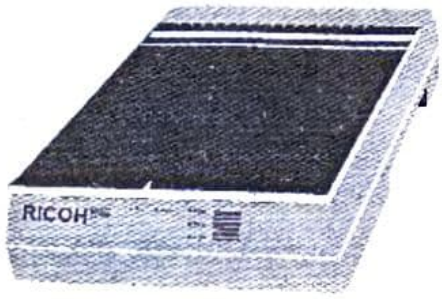
4)EGA monitors 5)Direct drive TTL monochrome

Composite monitors offer greater resolution than TV. They are low expensive.

RGB monitors offer better resolution for colour text than colour composite monitors. EGA (Enhanced Graphics Adapter) monitors are RGB monitors with higher resolution. TTL monitors are monochromatic and offer high-resolution steady images.



Monitor



Flat Bed Scanner

Fig. (4.3)

Video Board Characteristics:

Resolution

Resolution on a video display is the number of horizontal and vertical dots that make up the picture.e.g. basic resolution 640 x 480 means the image is 640 dots wide and 480 dots high or resolution means more dots on screen, which means video board needs more memory but results in excellent clarity.

In addition to resolution there are other characteristics like Character box size, monitors supported and graphics supported.

Video Monitor Characteristics

1) Dot pitch

More resolution means more dots on screen. The distance between the centers of the dots on the monitor is the monitor's dot pitch and it is measured in millimeter.

2) Horizontal Scan Frequency

Generally monitor works by directing a beam of electrons against inside of its screen. Phosphors inside screen become excitable and glow. So we can see image.

Horizontal scan frequency is the number of times that the beam sweeps horizontally per second. It is measured in Hz or Kilohertz. e.g. if basic VGA has 480 lines and there are 60 screens per second so 480 x 60 is 28,800 lines per second.

3) Multi-sync

The ability of monitor to handle multiple resolutions automatically is called multisyncing.

b) PRINTERS

Generally we classify o/p devices as hard copy and soft copy.
Printers, plotters are hard copy devices.
Printers are divided into basic two types:
1) Impact printer 2)Non-impact printer

In impact printers a mechanical contact between the print head and paper is carried out while in non-impact there is no such mechanism.

Impact Printers

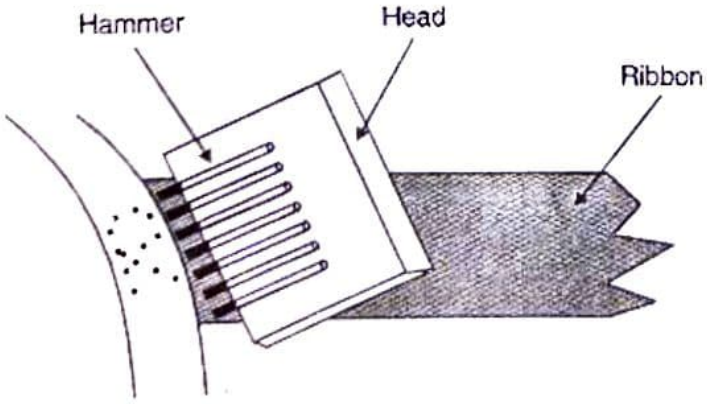
1) **Character Printers:** a) Dot matrix Printer b) Daisy wheel printer c) Golf ball Printer d) Cylinder printer
2) **Line printers:** a) Chain printer b) Band printer c) Drum printer

Non-Impact Printers

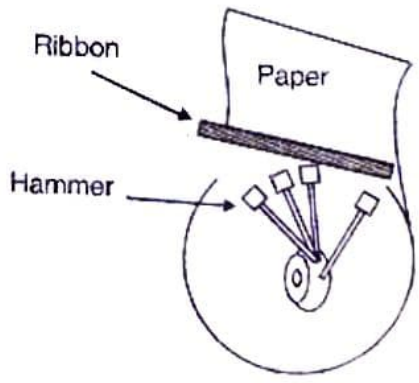
a) Thermal Printer b) Laser Printer c) Inkjet Printer d) Electrostatic Printer e) Electro-graphic Printer

1) Dot matrix printer:

It is an example of impact printer and very popular due to its low cost, high speed and NLQ (Near Letter Quality) print. Their speeds usually range between 80 & 600 cps. The printing head contains a vertical array of pins (refer fig.4.4). As the head moves across the paper, selected pins fire against an inked ribbon to form a pattern of dots on paper. Dot matrix printers universally bi-directional units as they print on both passes across the paper. They are capable of printing with maximum speed, but printing quality is not good like inkjet and available in black and white colour.



Dot Matrix Mechanism



Daisy Wheel Mechanism

Fig.(4.4)

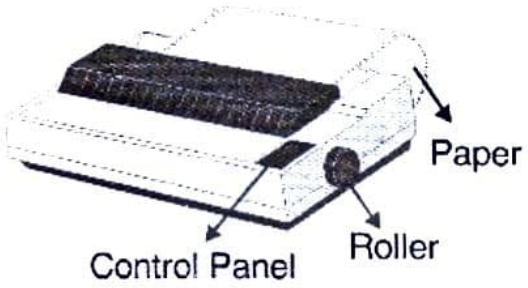
2) Daisy wheel printer:

It is so named because the print head is like a daisy flower, with printing arms appearing like petals of flower. It can operate at faster speeds up to 90 cps. The hub

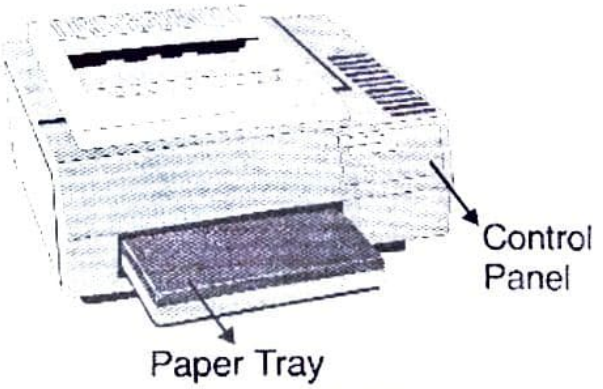
continuously rotating at higher speed and hammer strikes the appropriate character when it is in position just like a typewriter as shown in fig.(4.4).

3) Line printers

It prints whole line at a time but line printers of dot matrix are slow. Generally line printers use rapidly moving chains or bands of print characters or some form of print drums to print lines of information. It belongs to impact printer



Dot Matrix Printer



Laser Printer

4) Ink Jet Printer

An inkjet printer produces characters by spraying small ink droplets onto paper hence called non-impact printer.There is electrical generator, which causes piezoelectric crystal to vibrate and throws ink droplets supplied from reservoir.

Electrodes charge these and it is having vertical and horizontal deflecting plates in a manner to form desired characters on paper. It can have color printing. Inkjet printers provide a software through which user can arrange different properties like paper size, printing mode like Normal/Best/Draft.

2) Laser Printers

It is a high-speed high quality non-impact printer. It uses laser xerographic print mechanism. It uses a laser-produced light to charge photoconductive surface in the latent image of characters to be printed. In laser printer the image of the page to be printed is formed on a photosensitive drum. The photosensitive drum acts as an insulator in dark and when laser light beam falls on it then black part of drum gets charged. The charged image on drum is linked using black powder called "toner" and then electrostatically the image is transferred on a sheet of paper. Finally paper comes out the powdered image on paper is pressed with heat to make it permanent on paper.

Advantages:

i) It offers high quality printing in text and graphics.

ii) It offers wide selection of character fonts, printing quality mode under software control.

iii)Printing rate is highest it is 10 to 200 PPM (Pages Per min).

Disadvantages:

i) Initial Cost and cartridge cost is High.
ii) Multiple carbon copying is not possible.
iii) Colour printing is not possible.

4.3 I/O DEVICES AND DRIVES

In this topic we shall study other peripheral devices, which are used as secondary storage devices because they are auxiliary and provides extra storage capacity. Since they are auxiliary they need driving mechanism known as Drive and controllers.

a) Floppy disk and floppy Drive

Floppy disk is made up of round piece of plastic material coated with a magnetized recording material. Floppy disk is a Mylar disk with iron dioxide affixed on it. It has two sides, which require two head to read them. These two sides are called surfaces. Each surface is divided into 80 concentric circles called tracks. Each track is divided into sector. Each sector stores 512 bytes of information. Floppy is available in two sizes i.e. 1.2 MB (5.25 inch) and 1.44 MB (3.5 inch) as shown in fig.(4.5).

Advantages:

1. Direct information encoding onto disk.
2. Easily transportable and Relatively cheap
3. Multi purpose i.e., it is used for storage, input and output.
4. One can reuse many times like audiotape.

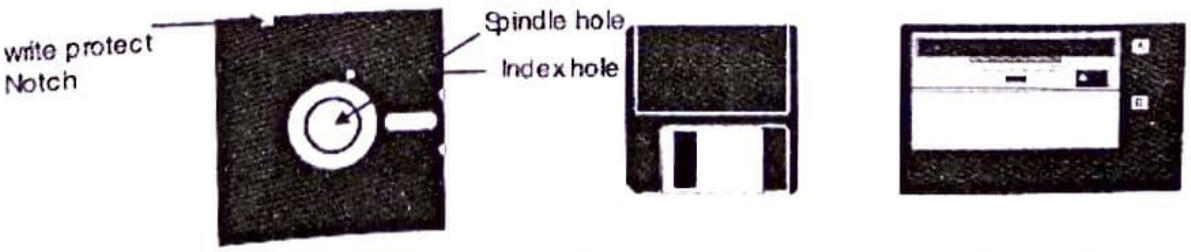
Disadvantages:

1. Floppy disks are to be handled carefully.
2. Adverse environmental conditions such as heat, dust, humidity etc easily damage them.

Floppy drive sub-system has following components,

1) Drive 2) Controller board 3) Cable connecting the drive to the controller

Power is connected to drive via smaller 4 pin Berg connector.



5 ¼" Floppy Disk 3 ½" Floppy Disk Floppy Drive

Fig. (4.5)

Controller

Whenever a computer wants to interface new device, it needs a controller to allow external device. XT type machine has separate controller board while AT type put floppy and hard disk controller function on same board.

b) Hard-Disk Drive

A hard-disk drive contains rigid metal or glass disks stacked up called plotters. Hard disk is fixed inside system i.e. it is sealed. But every hard disk structure is same as floppy disk like a floppy disk every side of each platter has read/write head. Hard disk can be considered as being made up multiple floppy disks put one above the other. All of the same tracks on each platter is collectively known as cylinder.

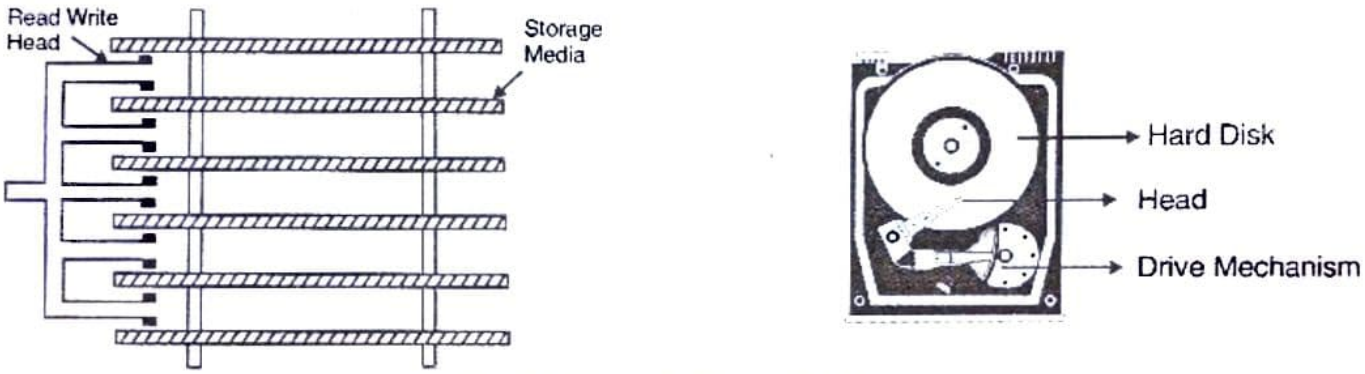


Fig. (4.6) Hard disk and Drive

Mostly hard disks have more read/write heads. These heads are attached to actuator arm. So when head 0 is positioned over track 142 on surface 0 by head actuator, head is also positioned on track 142 on surface.

To read a particular sector it must move read/write head over desired track and it must wait for the disk to rotate so that the desired sector is under the head and then must read.

Seek time

It is time to move to the particular cylinder.

Latency time

It is time to wait for the sector to rotate around and appear under the heads.

c) CD and CD-ROM Drive

Compact disks/CD's are optical storage media that are read with laser light. The surface of disk is arranged as a single spiral that begins inside the disk and travels towards the edge of disc. Compact disc define track by length of specific file. One file equals one track. They are laid sequentially on continuous spiral.

Generally, today all the software are stored on CD. Generally, CD-ROM drives can be connected to the rest of computer using IDE, SCSI, EIDE or parallel port. CDROM is a advance version of floppy disk having very large storage capacity but

commonly used for audio-video recording just like a audio or video cassette. The main feature of CD is it is recorded by a special technique and in the form of binary codes. The recording of audio songs, video films is done at the time of manufacturing by laser method the user can read it by means of CD player hence named as Read Only Memory (CDROM). CDs are used when the data or information is permanent used as reliable storage. The quality of CD is much excellent than a normal audio cassette but it is a costly device and delicate to handle.

There are different CD-ROM players and drives are available as follows,

1) CD-Recordable drives
2) Photo CD players
3) CD-Readable/writable drives
4) DVD (Digital Versatile Disk)
5) Computer based CD-ROM drives

Modems

A modem is a device containing both modulator and demodulator circuits. It is used to convert (modulate) binary signals into analog signal to transmit it over the telephone line at the receiver end the demodulator circuit converts these analog signals back into binary signals. Thus the modem is a technique of using telephone line for digital communications.

Use of Modem:

Let us see how modem is functioning for digital communication between the two computer terminals located at two distinct (long distance) places. Refer fig.(4.6) to understand its simple function

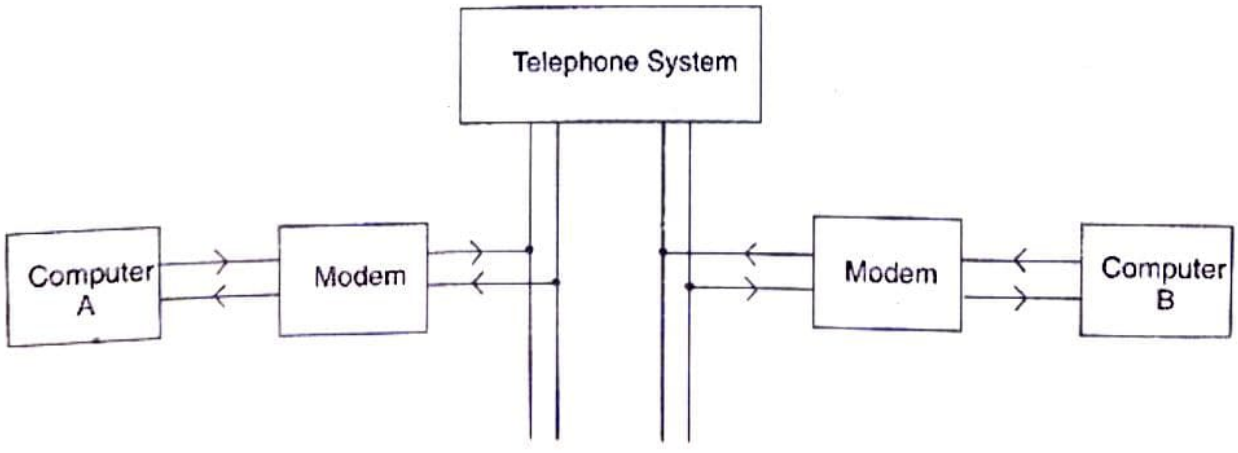


Fig. (4.6) Use of Modem

Digital communication between the two computers A and B is carried out in the following steps

1) The user at computer 'A' sends data in digital form.

2) The computer 'A' is modulating digital data into analog form; the modulation technique may be FSK type.
3) Modulated signal is within telephone frequency range it is transmitted over telephone line by selecting a proper dial code.
4) The computer ' B' terminal is selected by its dial code.
5) The modem of computer 'B' is converting analog modulated signal into original digital data by the process of demodulation.
6) The demodulated digital information is processed and displayed on the screen of terminal B.

The opposite and similar action is carried out while transmitting the data from Computer 'B' to Computer 'A'.

Multimedia Application

Multimedia is an entertataining audio-visual facility becoming very popular in computer world. It is a kit, which provides Audio mixing with images graphics and animation. Multimedia allows to run Audio/Video CDs and storage of this type of digital data with amplification facility and graphical display. Multimedia has variety of applications like video games, training tools etc.

Multimedia kit consists of

i) Sound Card ii) CDROM Drive iii) Standard Speaker system v)Microphone.

Sound Card Characteristics

Basically sound card is a part of Multimedia its quality depends on sound creation methods and resolution.

There are 8-bit, 16-bit, 32-bit cards. 8- bit are good for simple game sound and music, for good multimedia 16-bit is fine. 32-bit card is used for specialized application. Some sound card only plays MIDI files while some can playback only sampled file called wav files. But both capabilities are required for multimedia. Sound card provides:

i) An interface with PC
ii) Support for FM and MIDI files
iii) Sound Levels.

QUESTIONS

1.Select the correct alternative.

a) Floppy Disk is a ..... device

i) input ii)output iii)I/O

b) A standard Keyboard has .....Keys

i) 128 ii)101 iii)68

c) ..... is a multilayer disk.

i) Floppy disk ii)Compact disk iii)hard disk

d) Dot matrix printer is a ... ...Printer

i) Impact ii)non-impact iii)laser

e) To interconnect a PC with telephone line .....is necessary

i) scanner ii) controller iii)modem

f) In video colour monitors three basic colours are.....

i) RYB ii)RGB iii)GOB

g) For Running a video CD with PC ...is essential.

i) Controller ii) Multimedia Kit iii) Video card

h) A high quality high speed printer is..... printer

i) Inkjet ii)Laser iii)Dot matrix

Answer: a) I/O b) 101 c) hard disk d) Impact e) modem f) RGB g) Multimedia Kit h) Laser

2. List any four input and output devices.

3. Classify the following in to input and output devices. State their function in brief:

Mouse, Ink-Jet printer, CDROM, Modem, optical scanner.

4. Explain the difference between

a) Keyboard and mouse b) laser and Ink-jet printer

c) Floppy Disk and CDROM

5. Explain any one secondary storage device.

6. Enlist various printer types and explain any one.

7. Write advantages and disadvantages of Laser Printer.

8. Write a note on Hard disk and drive.

9. Explain the use of Modem.

10. Write the advantages and disadvantages of Floppy disk.

11. What is Multimedia and Sound card? Write their applications.

❑❑❑